



WAF for Frontend

WAF Introduction:

AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources.

AWS WAF gives you control over which traffic to allow or block to your web applications by defining customizable web security rules. You can use AWS WAF to create custom rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that are designed for your specific application. New rules can be deployed within minutes, letting you respond quickly to changing traffic patterns. Also, AWS WAF includes a full-featured API that you can use to automate the creation, deployment, and maintenance of web security rules.

With AWS WAF you pay only for what you use. AWS WAF pricing is based on how many rules you deploy and how many web requests your web application receives. There are no upfront commitments.

You can deploy AWS WAF on either Amazon CloudFront as part of your CDN solution, the Application Load Balancer (ALB) that fronts your web servers or origin servers running on EC2, or Amazon API Gateway for your APIs.

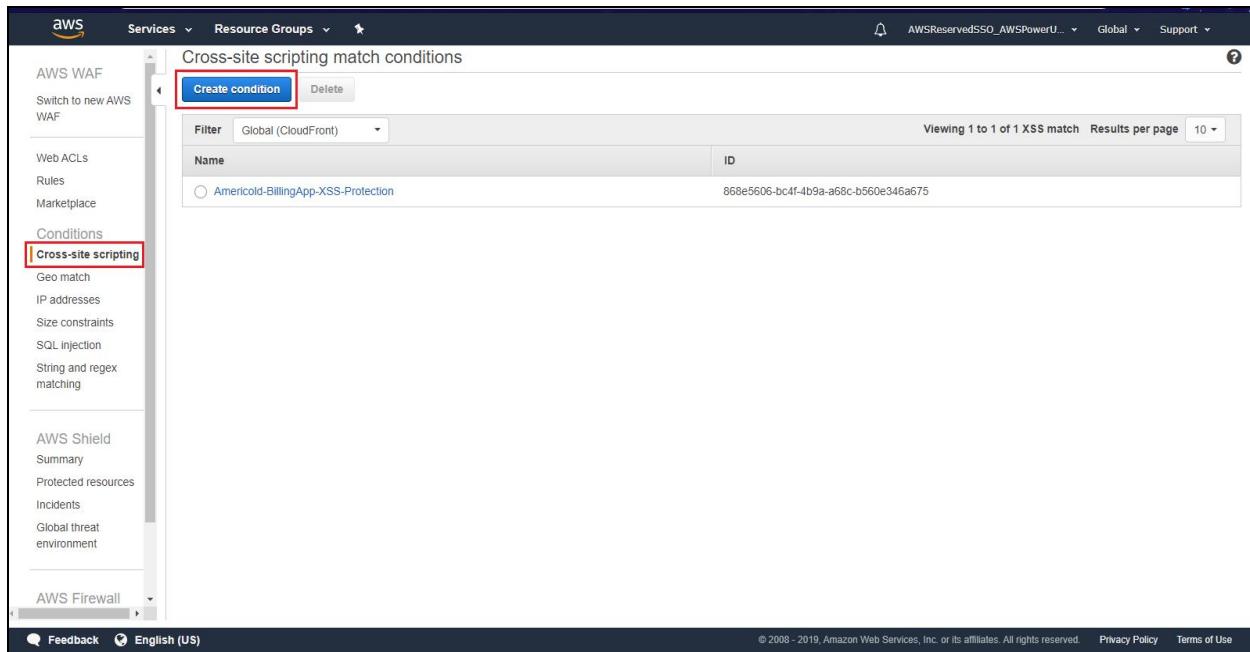
First, we have to login into the **AWS Console** and search for the WAF service.

The screenshot shows the AWS Console search interface. A red box highlights the search bar containing the text "waf". Below the search bar, the results list "WAF & Shield" as the top result, with the sub-description "Protects Against DDoS Attacks and Malicious Web Traffic". The search results are categorized by service type: Compute, Management Tools, Security, Identity & Compliance, Desktop & App St, WorkSpaces, AppStream 2.0, Internet Of Things, IoT Core, IoT 1-Click, IoT Device Manager, IoT Analytics, Greengrass, Amazon FreeRTOS, IoT Device Defender, Game Developme, and Amazon GameLift. Other visible categories include Storage, Media Services, and Database.

Following is the dashboard for AWS WAF. Now click on goto AWS WAF

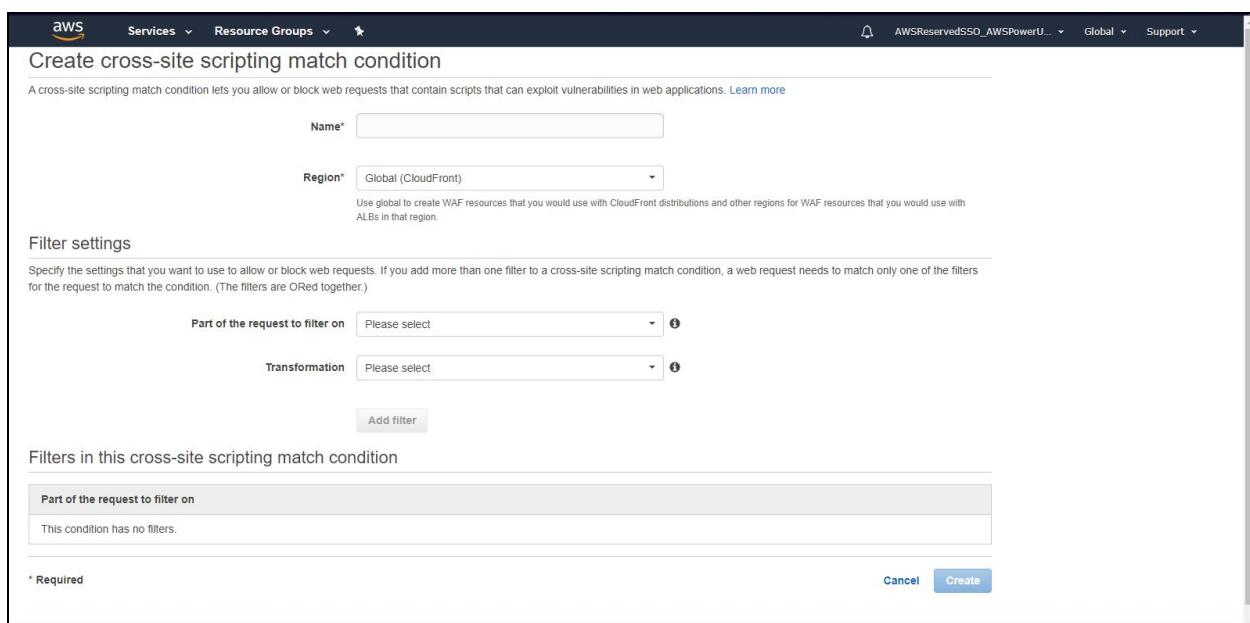
The screenshot shows the AWS WAF and AWS Shield dashboard. At the top center is a green downward-pointing arrow icon. Below it is the title "AWS WAF and AWS Shield" and a subtitle "AWS WAF and AWS Shield help protect your AWS resources from web exploits and DDoS attacks". The dashboard features three main sections: "AWS WAF" (represented by a cloud and wall icon), "AWS Shield" (represented by a shield icon), and "AWS Firewall Manager" (represented by a circular arrows icon). Each section has a brief description, a "Go to" button, and a "Learn more" link. The "Go to AWS WAF" button is highlighted with a red box.

Now we will be creating cross site scripting conditions so we will be selecting **Cross site scripting** and click on **Create Conditions** as shown below.



The screenshot shows the AWS WAF interface. On the left sidebar, under the 'Conditions' section, 'Cross-site scripting' is selected and highlighted with a red box. In the main content area, the title 'Cross-site scripting match conditions' is displayed above a table. The table has columns for 'Name' and 'ID'. One row is visible, labeled 'Americold-BillingApp-XSS-Protection' with ID '868e5606-bc4f-4b9a-a68c-b560e346a675'. At the top of the table, there is a 'Create condition' button, which is also highlighted with a red box. The top navigation bar includes links for AWSReservedSSO, AWSPowerU..., Global, and Support.

After clicking on create condition we will get the dashboard as follows to add the cross-site scripting conditions.



The screenshot shows the 'Create cross-site scripting match condition' wizard. The first step, 'Create cross-site scripting match condition', is completed. The second step, 'Configure cross-site scripting match condition', is currently being worked on. It requires a 'Name*' (which is empty) and a 'Region*' (set to 'Global (CloudFront)'). Below these, the 'Filter settings' section is expanded, showing fields for 'Part of the request to filter on' and 'Transformation', both currently set to 'Please select'. There is a 'Filters in this cross-site scripting match condition' section, which displays a message stating 'This condition has no filters.' At the bottom right, there are 'Cancel' and 'Create' buttons, with 'Create' being highlighted. A note at the bottom left indicates that the 'Part of the request to filter on' field is required.

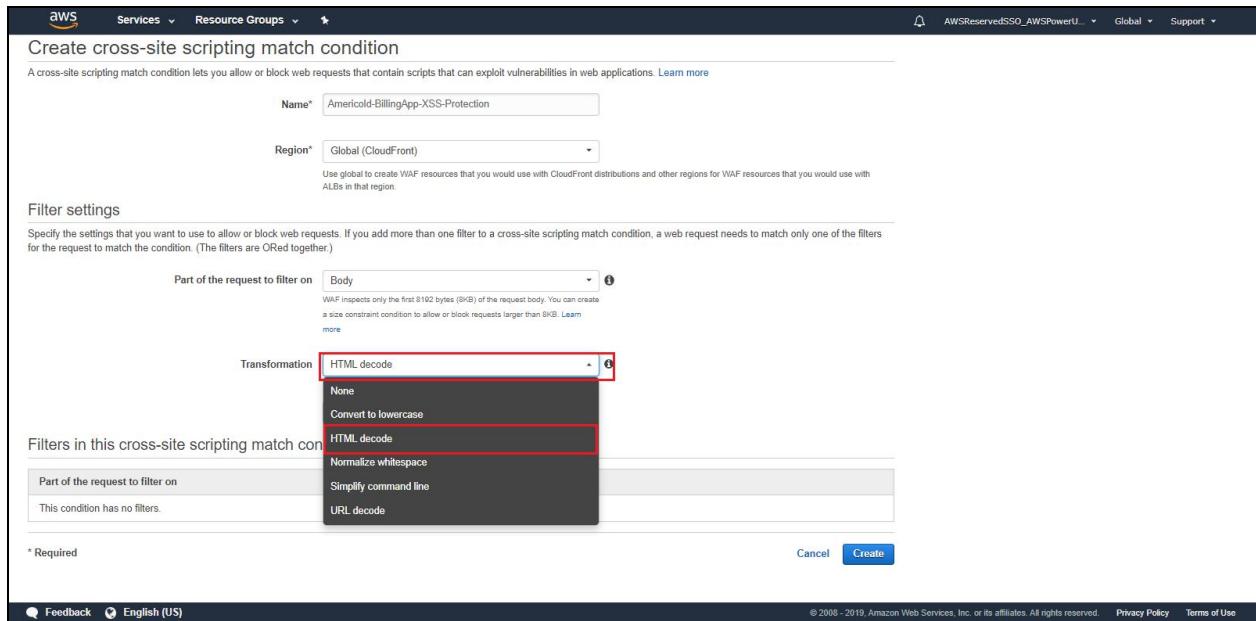
Here we gave the condition name as **Americold-BillingApp-XSS-Protection**.

The screenshot shows the 'Create cross-site scripting match condition' page in the AWS WAF console. The 'Name' field is filled with 'Americold-BillingApp-XSS-Protection'. The 'Region' dropdown is set to 'Global (CloudFront)'. Under 'Filter settings', there is a note about specifying filters to allow or block web requests. A dropdown menu for 'Part of the request to filter on' is open, showing options like Body, Header, HTTP method, Query string, Single query parameter (value only), All query parameters (values only), URI, and Body again. The 'Transformation' dropdown is set to 'HTML decode'. There is an 'Add filter' button. At the bottom, there are 'Cancel' and 'Create' buttons.

Now select **Body** in **Part of the request to filter on** to specify that where the condition have to apply

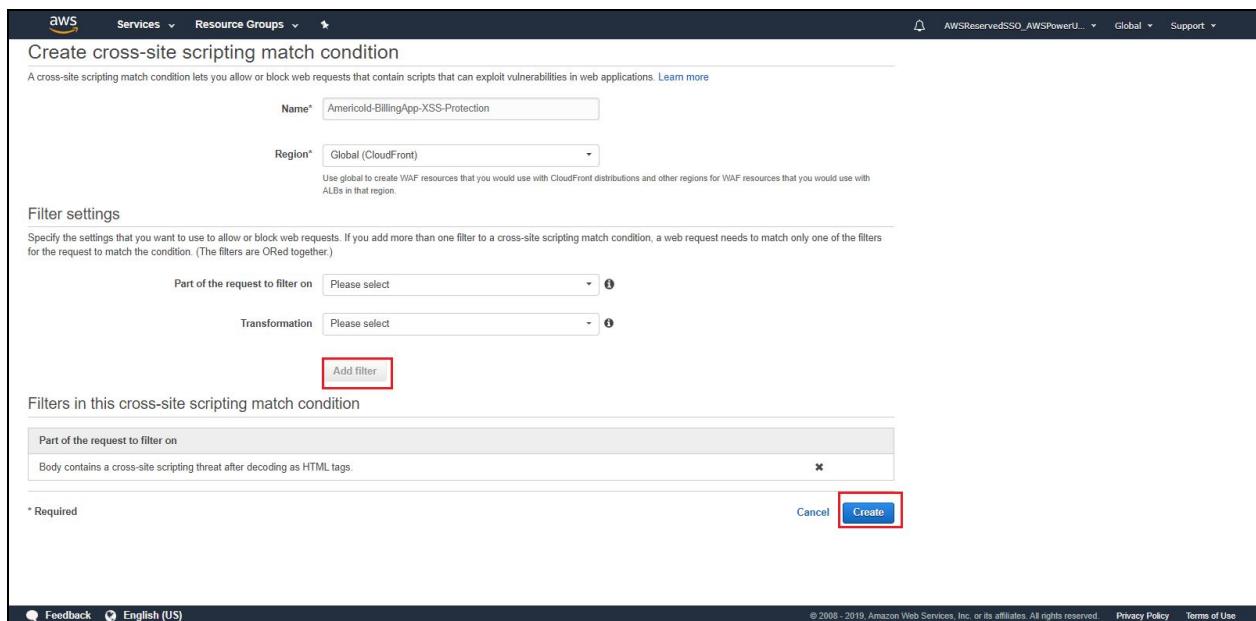
This screenshot is identical to the previous one, but the 'Body' option in the 'Part of the request to filter on' dropdown is highlighted with a red box. The rest of the interface is the same, including the 'Name' field ('Americold-BillingApp-XSS-Protection'), 'Region' ('Global (CloudFront)'), and the open 'Part of the request to filter on' dropdown.

Now select **HTML decode** in Transformation.



The screenshot shows the AWS WAF console interface for creating a new cross-site scripting match condition. The 'Name*' field is filled with 'Americold-BillingApp-XSS-Protection'. The 'Region*' dropdown is set to 'Global (CloudFront)'. In the 'Filter settings' section, there's a note about specifying filters to allow or block web requests. Below this, the 'Part of the request to filter on' dropdown is set to 'Body'. The 'Transformation' dropdown is open, showing several options: 'None', 'Convert to lowercase', 'HTML decode' (which is selected and highlighted with a red box), 'Normalize whitespace', 'Simplify command line', and 'URL decode'. At the bottom right, there are 'Cancel' and 'Create' buttons, with 'Create' being the active button.

Now click on the **Add filter** to add the filter.



This screenshot shows the same WAF configuration page after a filter has been added. The 'Add filter' button is highlighted with a red box. The 'Filters in this cross-site scripting match condition' section now displays a single filter: 'Body contains a cross-site scripting threat after decoding as HTML tags.' The 'Create' button at the bottom right is also highlighted with a red box.

We have to add all the filters that what we need and click on Create to create **Cross site scripting** as shown below.

Part of the request to filter on Please select Transformation Please select Add filter

Filters in this cross-site scripting match condition

Part of the request to filter on	X
Body contains a cross-site scripting threat after decoding as HTML tags.	X
Body contains a cross-site scripting threat after decoding as URL.	X
Header 'Cookie' contains a cross-site scripting threat after decoding as HTML tags.	X
Header 'Cookie' contains a cross-site scripting threat after decoding as URL.	X
Query string contains a cross-site scripting threat after decoding as HTML tags.	X
Query string contains a cross-site scripting threat after decoding as URL.	X
URI contains a cross-site scripting threat after decoding as HTML tags.	X
URI contains a cross-site scripting threat after decoding as URL.	X

* Required Cancel Create

Now go to **IP addresses** section and click on **create condition** to give the access for specific IP addresses. Here we are adding the **publicIP/CIDR** of Americold network which is provided by Americold networking team. It allows us to access the application with in the Americold network.

AWS Services Resource Groups IP match conditions AWSReservedSSO_AWSPowerU... Global Support ?

Create condition Delete

Name	ID
Americold_IP_Rule	6e254450-80b0-4132-b83c-c53e13230c94

Viewing 1 to 1 of 1 IP match Results per page 10 ▾

IP addresses

Size constraints SQL injection String and regex matching

AWS Shield Summary Protected resources Incidents Global threat environment

AWS Firewall Manager Security policies Rule groups Settings

Feedback English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Now we will get the dashboard as follows to configure the IP addresses

Create IP match condition

An IP match condition contains a list of IP addresses and/or IP address ranges. These IPs are the source of the requests that you want to allow or block. [Learn more](#)

Name*

Region* Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

IP addresses

Add one or more IP addresses or IP address ranges by using CIDR notation.

IP Version* IPv4 IPv6

Address* AWS WAF supports /8 or any range from /16 to /32 CIDR blocks for IPv4 Examples:
For a single IP address, please specify like 192.0.2.44/32
For an IP range from 192.0.2.0 to 192.0.2.255, please use 192.0.2.0/24

Add IP address or range

Filters in IP match condition

IP address of the request to filter on
This condition has no filters.

* Required

Now we configured condition name as **Americold_IP_Rule** and give the IP address and click on **Add IP address or range**.

Create IP match condition

An IP match condition contains a list of IP addresses and/or IP address ranges. These IPs are the source of the requests that you want to allow or block. [Learn more](#)

Name*

Region* Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

IP addresses

Add one or more IP addresses or IP address ranges by using CIDR notation.

IP Version* IPv4 IPv6

Address* AWS WAF supports /8 or any range from /16 to /32 CIDR blocks for IPv4 Examples:
For a single IP address, please specify like 192.0.2.44/32
For an IP range from 192.0.2.0 to 192.0.2.255, please use 192.0.2.0/24

Add IP address or range

Filters in IP match condition

IP address of the request to filter on
This condition has no filters.

* Required

Now the IP address will be added to the **Filter in IP match condition** and then click on **create**. In the same way that we have mentioned above we have to add all IP addresses that can access our application

A **size constraint** statement compares a number of bytes against the size of a request component, using a comparison operator, such as greater than (>) or less than (<). If the size of the string not satisfies the condition we can restrict the access.

Now go to **Size constraints** and click on **Create Conditions**.

Now select **URI** in **Part of the request to filter on** to specify that where the condition have to apply

The screenshot shows the 'Create size constraint condition' page in the AWS WAF console. The 'Name' field is set to 'Americold_SizeConstraint'. The 'Region' is set to 'Global (CloudFront)'. In the 'Filter settings' section, the 'Part of the request to filter on' dropdown is open, showing options like 'Header', 'HTTP method', 'Query string', 'Single query parameter (value only)', 'All query parameters (values only)', 'URI', and 'Body'. The 'URI' option is highlighted with a red box. Below this, there's a note about specifying filters for requests to match the condition. At the bottom, there are 'Cancel' and 'Create' buttons.

Now select **Greater than** in **Comparison operator**.

The screenshot shows the same 'Create size constraint condition' page. The 'Part of the request to filter on' is still set to 'URI'. In the 'Comparison operator' dropdown, the options are 'Please select', 'Equals', 'Not equal', 'Greater than', 'Greater than or equal', 'Less than', and 'Less than or equal'. The 'Greater than' option is highlighted with a red box. The rest of the page remains consistent with the first screenshot, including the note about filters and the 'Create' button at the bottom.

Now we gave the name as **Size Constraint**, Region will be as default, select **URI** in the part of the request to filter on, select **Greater than** in the comparison operator, give the size as **100000000** and select **None** in the Transformation then click on the **Add filter** button.

Create size constraint condition

A size constraint condition lets you allow or block web requests based on the lengths of specified parts of the request. [Learn more](#)

Name*

Region*

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a size constraint condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on

Comparison operator

Size (Bytes)

Transformation

Add filter

Filters in this size constraint condition

Part of the request to filter on

This condition has no filters.

* Required Cancel Create

Now the Filter will be added and click on **create** button.

Create size constraint condition

A size constraint condition lets you allow or block web requests based on the lengths of specified parts of the request. [Learn more](#)

Name*

Region*

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a size constraint condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on

Comparison operator

Size (Bytes)

Transformation

Add filter

Filters in this size constraint condition

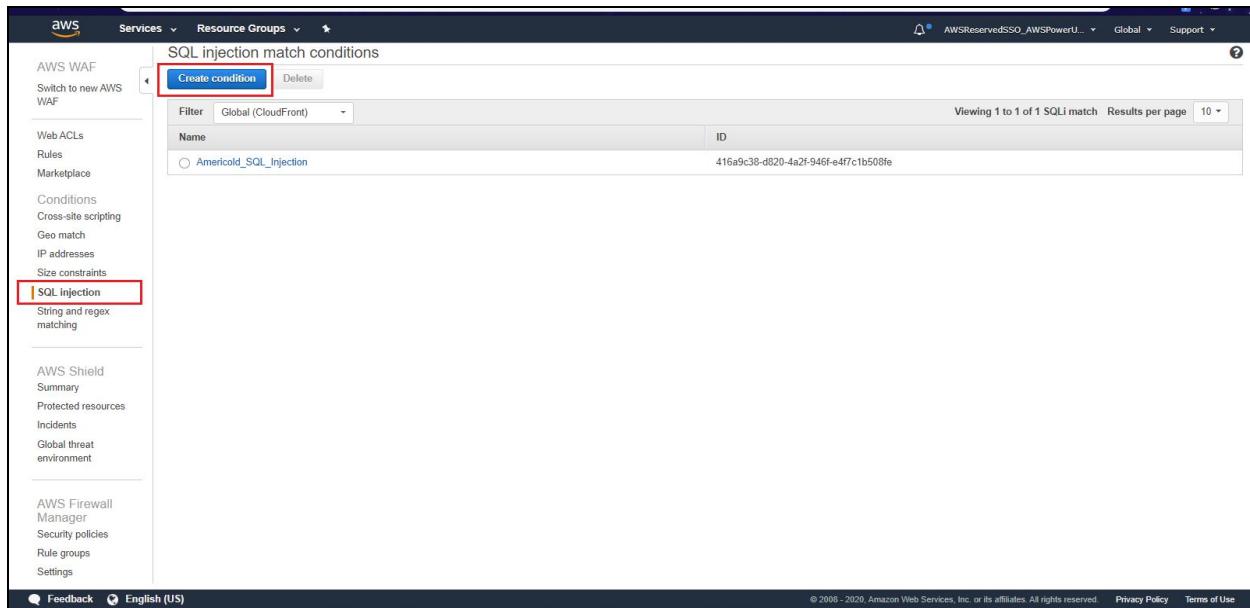
Part of the request to filter on

The length of the URI is greater than 100000000. X

* Required Cancel Create

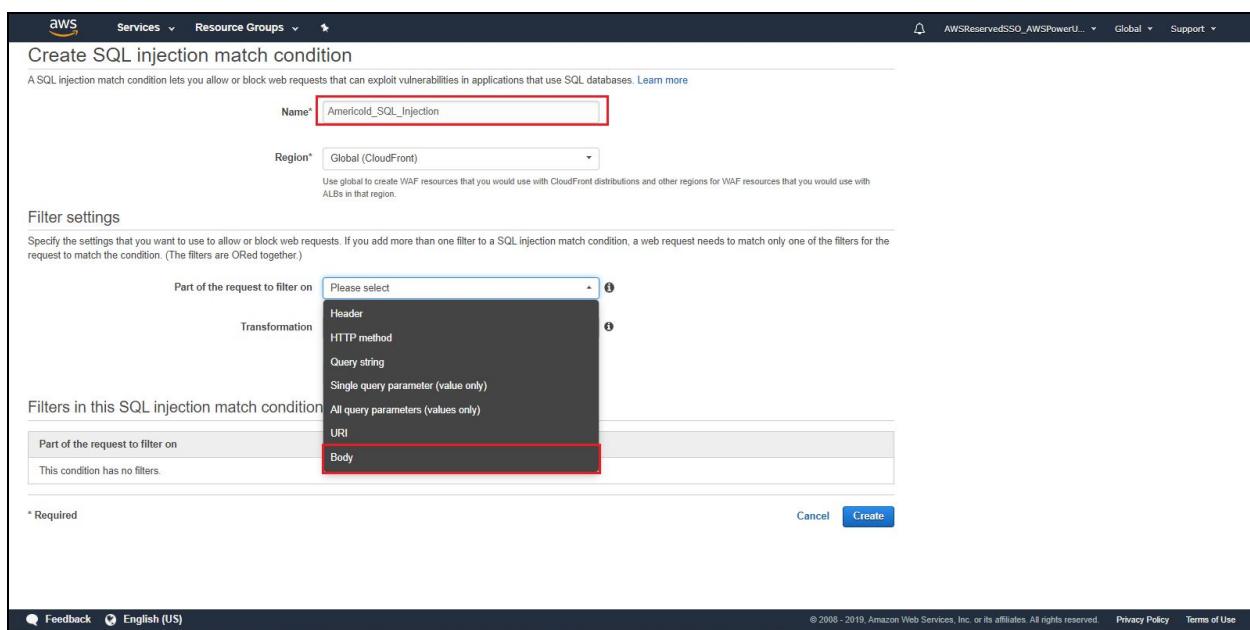
SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. To restrict these types of vulnerabilities we have added SQL injection conditions.

Now go to **SQL Injections** and click on **create conditions** to add the **SQL Injection condition**.



The screenshot shows the AWS WAF console. On the left sidebar, under the 'SQL injection' section, the 'String and regex matching' option is selected and highlighted with a red box. In the main content area, there is a table titled 'SQL injection match conditions'. A single row is listed with the name 'Americold_SQL_Injection' and an ID of '416a9c38-d820-4a2f-946f-e4f7c1b508fe'. At the top of the table, there is a 'Create condition' button, which is also highlighted with a red box.

Now select **body** in **Part of the request to filter on** to specify that where the condition have to apply



The screenshot shows the 'Create SQL injection match condition' wizard. The 'Name*' field contains 'Americold_SQL_Injection' and is highlighted with a red box. The 'Region*' dropdown is set to 'Global (CloudFront)'. Under 'Filter settings', there is a dropdown menu for 'Part of the request to filter on' with options: 'Header', 'HTTP method', 'Query string', 'Single query parameter (value only)', 'All query parameters (values only)', and 'URI'. The 'Body' option is highlighted with a red box. Below this, there is a table for 'Filters in this SQL injection match condition' with a single row showing 'Part of the request to filter on: Body'. The 'Create' button at the bottom right is also highlighted with a red box.

Now select **HTML decode** in Transformation.

The screenshot shows the 'Create SQL injection match condition' page in the AWS WAF console. The 'Name*' field is set to 'Americold_SQL_Injection'. The 'Region*' field is set to 'Global (CloudFront)'. Under 'Filter settings', there is a dropdown for 'Part of the request to filter on' set to 'Body'. A dropdown for 'Transformation' is open, showing options: 'Please select', 'None', 'Convert to lowercase', 'HTML decode' (which is highlighted with a red box), 'Normalize whitespace', 'Simplify command line', and 'URL decode'. Below the dropdown, the 'Filters in this SQL injection match condition' section shows a single filter entry: 'Part of the request to filter on: Body' with the note 'Body contains SQL injection threat after decoding as HTML tags.' At the bottom right are 'Cancel' and 'Create' buttons.

Now the filter will be added as shown below

The screenshot shows the same 'Create SQL injection match condition' page after adding a filter. The 'Transformation' dropdown now shows 'HTML decode'. Below it, a 'Add filter' button is visible. The 'Filters in this SQL injection match condition' section now contains two entries: 'Part of the request to filter on: Body' with the note 'Body contains SQL injection threat after decoding as HTML tags.' and a second entry 'Part of the request to filter on: None'. At the bottom right are 'Cancel' and 'Create' buttons.

In the same way we will add all filters as per the requirement and click on **create** button to confirm.

Region* Global (CloudFront)
Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings
Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a SQL injection match condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on: URI
Transformation: URL decode
Add filter

Filters in this SQL injection match condition

Part of the request to filter on
Body contains SQL injection threat after decoding as HTML tags.
Body contains SQL injection threat after decoding as URL.
Header 'Cookie' contains SQL injection threat after decoding as HTML tags.
Header 'Cookie' contains SQL injection threat after decoding as URL.
Query string contains SQL injection threat after decoding as HTML tags.
Query string contains SQL injection threat after decoding as URL.
URI contains SQL injection threat after decoding as HTML tags.
URI contains SQL injection threat after decoding as URL.

* Required Cancel **Create**

Feedback English (US) © 2006 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

If you want to allow or block web requests based on strings that match a regular expression (regex) pattern that appears in the requests we need to create String and regex match conditions.

Now go to **String and regex matching** and click on **Create condition**.

AWS WAF
Switch to new AWS WAF

Web ACLs
Rules
Marketplace
Conditions
Cross-site scripting
Geo match
IP addresses
Size constraints
SQL injection
String and regex matching

String and regex match conditions
Create condition Delete View regex pattern sets

Filter Global (CloudFront)

Name	Type	ID
Americold_StringMatchCondition_2	String match	24ccce6a-ae59-4e1d-a689-e62f6d636117
Americold_StringMatchCondition	String match	8a069986-f5f9-47a3-a2ca-8db01a55227
Americold_StringMatchCondition_1	String match	46eec873-5053-4b20-ab9b-84c7b8a71e2e

Viewing 1 to 3 of 3 String match Results per page 10

AWS Shield
Summary
Protected resources
Incidents
Global threat environment

AWS Firewall Manager
Security policies
Rule groups
Settings

Feedback English (US) © 2006 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Now configure the name as **BlackList1**, configure the **Type** as **String match** and in **filter settings** section Select **Body** in Part of the request to filter on, **Contains** in match type, select **HTML decode** in the Transformation and some value to match then click on the **Add filter**.

After adding all the filters click on **create** button to confirm.

Following is the output for the above steps.

The screenshot shows the AWS WAF console under the Services menu. On the left, there's a sidebar with options like AWS WAF, Web ACLs, Rules, Marketplace, Conditions, AWS Shield, and AWS Firewall Manager. The 'String and regex matching' section is currently selected. In the main pane, it says 'String and regex match conditions' and shows a table with two rows:

Name	Type
Americold_StringMatchCondition_2	String
Americold_StringMatchCondition	String
Americold_StringMatchCondition_1	String

To the right of the table is a detailed view for the selected condition, titled 'Americold_StringMatchCondition'. It includes tabs for 'Add filter' and 'Delete filter'. A 'Filter' section is expanded, showing a list of options:

- Body contains: "" after decoding as HTML tags.
- Body contains: "alert()" after decoding as HTML tags.
- Body contains: "</script>" after decoding as HTML tags.
- Body contains: "<script>" after decoding as HTML tags.
- Body contains: "window".
- Body contains: "embed".
- Body contains: "onabort".
- Body contains: "javascript".
- Body contains: "onblur".
- Body contains: "onchange".

Here we have created another condition in **string and regex matching** as **Americold_StringMatchCondition_1**.

This screenshot is similar to the previous one, showing the AWS WAF console with the 'String and regex matching' section selected. The table now has three rows:

Name	Type
Americold_StringMatchCondition_2	String
Americold_StringMatchCondition	String
Americold_StringMatchCondition_1	String

The detailed view for 'Americold_StringMatchCondition_1' is shown on the right, with the 'Filter' section expanded. The list of options is identical to the one in the previous screenshot:

- Body contains: "onclick" after converting to lowercase.
- Body contains: "onfocus" after converting to lowercase.
- Body contains: "onmousedown" after converting to lowercase.
- Body contains: "ondblclick" after converting to lowercase.
- Body contains: "onkeydown" after converting to lowercase.
- Body contains: "onkeyup" after converting to lowercase.
- Body contains: "onerror" after converting to lowercase.
- Body contains: "onmouseout" after converting to lowercase.
- Body contains: "onkeypress" after converting to lowercase.

Here we created another condition in string and regex matching as **Americold_StringMatchCondition_2**.

The screenshot shows the AWS WAF console. On the left sidebar, under 'String and regex matching', the 'Rules' option is selected. In the main content area, the 'String and regex match conditions' section is displayed. A 'Create condition' button is visible. Below it, a table lists three conditions: 'Americold_StringMatchCondition_2' (selected), 'Americold_StringMatchCondition', and 'Americold_StringMatchCondition_1'. The 'Americold_StringMatchCondition_2' row is highlighted with a red box. To the right, a detailed view of this condition is shown in a modal window titled 'Americold_StringMatchCondition_2'. This view includes a 'Filter' section with various regex patterns for 'Body contains' such as 'onreset', 'document', 'onresize', etc., all of which are enclosed in a red box.

Now go to **Rules section** and click on the **Create Rule**.

The screenshot shows the AWS WAF console. On the left sidebar, under 'String and regex matching', the 'Rules' option is selected. In the main content area, the 'Rules' section is displayed. A 'Create rule' button is highlighted with a red box. Below it, a table lists six existing rules: 'Americold_SQL_Injection_Rule', 'Americold_Size_Constraint_Rule', 'Americold_String_Match_Rule', 'Americold_XSS_Rule', 'Americold_Backend_Rule', and 'Americold_IP_Rule'. The table includes columns for 'Name', 'Type', and 'ID'. The entire 'Create rule' button is enclosed in a red box.

Now configure the name as **Americold_IP_Rule**, CloudWatch metric name will be the same as the name of the rule, Select Role Type as **Regular Role** and region as default. Now coming to **Add conditions** section and add the **conditions** as follows.

The screenshot shows the 'Create rule' page in the AWS WAF console. The 'Name' field is set to 'Americold_IP_Rule'. The 'CloudWatch metric name' field is also set to 'AmericoldIPRule'. The 'Rule type' is selected as 'Regular rule'. The 'Region' is set to 'Global (CloudFront)'. Below these fields, there is a note about using global resources with CloudFront distributions and other regions for WAF resources. The 'Add conditions' section is expanded, showing a modal dialog titled 'When a request' with the condition 'does not originate from an IP address in'. A dropdown menu shows 'Americold_IP_Rule' selected. Below this, a list of IP addresses is shown: '12.150.73.2/32' and '12.198.28.194/32'. An 'Add condition' button is visible at the bottom left of the modal. At the bottom right of the modal are 'Cancel' and 'Create' buttons. A red box highlights the 'Create' button. The main page has a note about required fields and links to Feedback, English (US), Privacy Policy, and Terms of Use.

Following is the rules for **Americold_SQL_Injection_Rule**.

The screenshot shows the 'Rules' list in the AWS WAF console. On the left, the navigation pane includes 'AWS WAF', 'Web ACLs', 'Rules' (which is selected), 'Conditions', 'Cross-site scripting', 'Geo match', 'IP addresses', 'Size constraints', 'SQL injection', 'String and regex matching', 'AWS Shield', 'AWS Firewall'. The 'Rules' list shows several rules, with 'Americold_SQL_Injection_Rule' selected and highlighted by a red box. The 'Type' column indicates they are all 'Regular'. To the right, the detailed view for 'Americold_SQL_Injection_Rule' is shown. It includes an 'Edit rule' button and a 'Filters in Americold_SQL_Injection' section. This section lists various filters: 'URI contains SQL injection threat after decoding as URL', 'Header 'cookie' contains SQL injection threat after decoding as HTML tags', 'Query string contains SQL injection threat after decoding as HTML tags', 'Body contains SQL injection threat after decoding as URL', 'Header 'cookie' contains SQL injection threat after decoding as URL', 'URI contains SQL injection threat after decoding as HTML tags', 'Query string contains SQL injection threat after decoding as URL', and 'Body contains SQL injection threat after decoding as HTML tags'. A red box highlights the entire 'Filters in Americold_SQL_Injection' section. The bottom of the page includes a note about required fields and links to Feedback, English (US), Privacy Policy, and Terms of Use.

Following are the rules for **Americold_Size_Constraint_Rule**.

The screenshot shows the AWS WAF Rules page. On the left sidebar, under the 'Rules' section, 'Americold_Size_Constraint_Rule' is selected and highlighted with a red box. The main pane displays a table of rules with the following data:

Name	Type
Americold_SQL_Injection_Rule	Regular
Americold_Size_Constraint_Rule	Regular
Americold_String_Match_Rule	Regular
Americold_XSS_Rule	Regular
Americold_Backend_Rule	Regular
Americold_IP_Rule	Regular

To the right, the details for the selected rule are shown in a modal window titled 'Americold_Size_Constraint_Rule'. The modal contains the following text:

When a request matches at least one of the filters in the size constraint condition Americold_SizeConstraint

Filters in Americold_SizeConstraint

The length of the URI is greater than 10000000.

Following are the rules for **Americold_String_Match_Rule**.

The screenshot shows the AWS WAF Rules page. On the left sidebar, under the 'Rules' section, 'Americold_String_Match_Rule' is selected and highlighted with a red box. The main pane displays a table of rules with the following data:

Name	Type
Americold_SQL_Injection_Rule	Regular
Americold_Size_Constraint_Rule	Regular
Americold_String_Match_Rule	Regular
Americold_XSS_Rule	Regular
Americold_Backend_Rule	Regular
Americold_IP_Rule	Regular

To the right, the details for the selected rule are shown in a modal window titled 'Americold_String_Match_Rule'. The modal contains the following text:

When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition_2

Filters in Americold_StringMatchCondition_2

Body contains: "onreset".
Body contains: "document".
Body contains: "onresize".
Body contains: "onunload".
Body contains: "onselect".
Body contains: "behavior".
Body contains: "onsubmit".
Body contains: "onmousemove".
Body contains: "onmouseup".
Body contains: "expression".

And

When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition

Filters in Americold_StringMatchCondition

Body contains: "onabort".
Body contains: "onchange".

AWS Shield	Body contains: "onmouseup". Body contains: "expression".
Summary	
Protected resources	
Incidents	
Global threat environment	
AWS Firewall Manager	
Security policies	
Rule groups	
Settings	

And

When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition
Filters in Americold_StringMatchCondition
Body contains: "onabort".
Body contains: "onchange".
Body contains: "onembed".
Body contains: "</script>" after decoding as HTML tags.
Body contains: "javascript".
Body contains: "onblur".
Body contains: "<script>" after decoding as HTML tags.
Body contains: ":" after decoding as HTML tags.
Body contains: "alert()" after decoding as HTML tags.
Body contains: "window".

And

When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition_1
Filters in Americold_StringMatchCondition_1
Body contains: "onfocus" after converting to lowercase.
Body contains: "onclick" after converting to lowercase.
Body contains: "onmousedown" after converting to lowercase.

Role groups	Body contains: "</script>" after decoding as HTML tags. Body contains: "javascript". Body contains: "onblur". Body contains: "<script>" after decoding as HTML tags. Body contains: ":" after decoding as HTML tags. Body contains: "alert()" after decoding as HTML tags. Body contains: "window".
Settings	

And

When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition_1
Filters in Americold_StringMatchCondition_1
Body contains: "onfocus" after converting to lowercase.
Body contains: "onclick" after converting to lowercase.
Body contains: "onmousedown" after converting to lowercase.
Body contains: "onmouseout" after converting to lowercase.
Body contains: "onkeydown" after converting to lowercase.
Body contains: "onerror" after converting to lowercase.
Body contains: "onkeyup" after converting to lowercase.
Body contains: "ondblclick" after converting to lowercase.
Body contains: "onkeypress" after converting to lowercase.

Following are the rules for **Americold_XSS_Rule**.

The screenshot shows the AWS WAF Rules page. On the left sidebar, under the 'Rules' section, 'Americold_XSS_Rule' is selected and highlighted with a red box. The main content area displays the rule configuration for 'Americold_XSS_Rule'. The rule is defined under the 'Cross-site scripting' match condition 'Americold-BillingApp-XSS-Protection'. The filters listed include:

- URI contains a cross-site scripting threat after decoding as URL.
- URI contains a cross-site scripting threat after decoding as HTML tags.
- Header 'cookie' contains a cross-site scripting threat after decoding as URL.
- Header 'cookie' contains a cross-site scripting threat after decoding as HTML tags.
- Body contains a cross-site scripting threat after decoding as URL.
- Body contains a cross-site scripting threat after decoding as HTML tags.
- Query string contains a cross-site scripting threat after decoding as URL.
- Query string contains a cross-site scripting threat after decoding as HTML tags.

Following are the rules for **Americold_Backend_Rule**.

The screenshot shows the AWS WAF Rules page. On the left sidebar, under the 'Rules' section, 'Americold_Backend_Rule' is selected and highlighted with a red box. The main content area displays the rule configuration for 'Americold_Backend_Rule'. The rule is defined under the 'String match' condition 'Americold_StringMatchCondition'. The filters listed include:

- Body contains: "javascript".
- Body contains: "</script>" after decoding as HTML tags.
- Body contains: "<script>" after decoding as HTML tags.
- Body contains: "embed".
- Body contains: "onabort".
- Body contains: "alert()" after decoding as HTML tags.
- Body contains: "window".
- Body contains: ":" after decoding as HTML tags.
- Body contains: "onblur".
- Body contains: "onchange".

Now we have to go to **WebACLS** portal and click on **Configure web ACL**.

The screenshot shows the AWS WAF Web ACls page. On the left, there's a sidebar with options like 'Web ACls' (which is selected and highlighted with a red box), 'Rules', 'Marketplace', 'Conditions', 'Cross-site scripting', 'Geo match', 'IP addresses', 'Size constraints', 'SQL injection', 'String and regex matching', 'AWS Shield', 'Summary', 'Protected resources', 'Incidents', 'Global threat environment', and 'AWS Firewall Manager'. The main area is titled 'Web ACls' and contains a table with one row. The table has columns for 'Name' and 'ID'. The 'Name' column contains 'Americold_UI_Dev' and the 'ID' column contains '56acef5c-6610-43fc-a7c2-45397b6eb4c9'. A blue button labeled 'Create web ACL' is visible at the top left of the main area.

Now give a name to web ACL as **Americold_UI_Dev** and CloudWatch metric name will be automatically generated. Click on **Next**.

The screenshot shows the 'Set up a web access control list (web ACL)' wizard, specifically Step 1: Name web ACL. On the left, there's a sidebar with 'Concepts overview' and steps: 'Step 1: Name web ACL' (selected and highlighted with a red box), 'Step 2: Create conditions', 'Step 3: Create rules', and 'Step 4: Review and create'. The main area is titled 'Name web ACL' and contains fields for 'Web ACL name*' (set to 'Americold_UI_Dev'), 'CloudWatch metric name*' (set to 'AmericoldUIDev'), 'Region*' (set to 'Global (CloudFront)'), and 'AWS resource to associate' (a dropdown menu showing 'Select a resource' and an option 'E3GILM5AHJUTX5 - testinventoryrev...'). At the bottom, there are buttons for 'Cancel', 'Previous', and 'Next' (highlighted with a red box). A note at the bottom right says: 'You can associate this web ACL with more resources after you finish the wizard. On the Web ACls page for this web ACL, see the Rules tab.'

Now in the second step create conditions for Cross-site scripting match conditions and now click on **create condition**.

Set up a web access control list (web ACL)

Step 2: Create conditions

Cross-site scripting match conditions

Name: Americold-BillingApp-XSS-Protection

A cross-site scripting match condition specifies the parts of a web request (such as a User-Agent header) that you want AWS WAF to inspect for cross-site scripting threats. [Learn more](#)

Geo match conditions

Name: Americold_IP_Rule

You don't have any geo match conditions. Choose [Create condition](#) to get started.

A geo match condition lets you allow, block, or count web requests based on the geographic origin of the request. [Learn more](#)

IP match conditions

Name: Americold_SQL_Injection

An IP match condition specifies the IP addresses and/or IP address ranges that you want to use to control access to your

Concepts overview

Web ACL example if requests match

Rule 1, Bad User-Agents, then block

IP match condition Suspicious IPs

and

String match condition Bad bots

or if requests match

Rule 2, Detect SQLI, then block

SQL injection match condition SQLI checks

otherwise, perform the default action

Default action

Allow requests that don't match any rules

Click on **Next** button.

SQL injection match conditions

Name: Americold_SQL_Injection

A SQL injection match condition specifies the parts of a web request (such as a User-Agent header) that you want AWS WAF to compare to a set size. [Learn more](#)

String and regex match conditions

Name: Americold_StringMatchCondition_2

String Match

Name: Americold_StringMatchCondition

String Match

Name: Americold_StringMatchCondition_1

String Match

A string match condition, or a regex match condition, specifies the part of a web request (such as a User-Agent header) and the text (the value of the header) that you want to use to control access to your content. Create separate conditions for strings or regex patterns that you want to allow or block. [Learn more](#)

Cancel Previous Next

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Now go to step 3 and create rules for web ACL.

The screenshot shows the 'Create rules' step of the web ACL setup process. On the left, a sidebar lists steps: Step 1: Name web ACL, Step 2: Create conditions, Step 3: Create rules (highlighted with a red box), and Step 4: Review and create. The main content area is titled 'Create rules' and contains two sections: 'Add rules to a web ACL' and 'Concepts overview'. The 'Add rules to a web ACL' section includes a 'Rules' dropdown menu (set to 'Select a rule'), a 'Create rule' button, and a table for defining rules. The 'Concepts overview' section provides examples of rules and actions:

- Rule 1, Bad User-Agents, then block**: IP match condition (Suspicious IPs) and String match condition (Bad bots).
- Rule 2, Detect SQLI, then block**: SQL injection match condition (SQLI checks).
- otherwise, perform the default action: Allow requests that don't match any rules.

At the bottom, there are 'Feedback', 'English (US)', and navigation buttons for 'Cancel', 'Previous', and 'Review and create'.

Select Rules that was created.

The screenshot shows the 'Create rules' step with a specific rule selected. The 'Rules' dropdown menu is open, showing a list of self-owned rules: Americold_SQL_Injection, Americold_Size_Condition, Americold_String_Match, Americold_XSS_Rule, Americold_Banned_UA, and Americold_IP_Rule. The 'Americold_SQL_Injection' rule is highlighted with a red box. The rest of the interface is identical to the previous screenshot, including the 'Concepts overview' sidebar and the 'Review and create' button at the bottom.

Here we selected the check boxes in the rules as Block and check any Default action and click on review and create

Step 1. Name web ACL
Step 2. Create conditions
Step 3: Create rules
Step 4. Review and create

Rules contain the conditions that you want to use to filter web requests. You add rules to a web ACL, and then specify whether you want to allow or block requests based on each rule. Learn more

Add rules to a web ACL

Rules **Americold_IP_Rule** Add another rule Create rule

If a request matches all of the conditions in a rule, take the corresponding action

Order	Rule	Action
1	Americold_IP_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>
2	Americold_SQL_Injection_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>
3	Americold_Size_Constraint_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>
4	Americold_String_Match_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>
5	Americold_XSS_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>
6	Americold_Backend_Rule	<input type="radio"/> Allow <input checked="" type="radio"/> Block <input type="radio"/> Count <input type="checkbox"/>

If a request doesn't match any rules, take the default action

Default action* Allow all requests that don't match any rules Block all requests that don't match any rules

Web ACL example if requests match

Rule 1, Bad User-Agents, then block

IP match condition Suspicious IPs

and

String match condition Bad bots

or if requests match

Rule 2, Detect SQLI, then block

SQL injection match condition SQLI checks

otherwise, perform the default action

Default action

Allow requests that don't match any rules

* Required Cancel Previous **Review and create**

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Now we will get into the step4, Click on Confirm and Create.

Concepts overview
Step 1: Name web ACL
Step 2: Create conditions
Step 3: Create rules
Step 4: Review and create

Review and create

Review your settings, and then choose Confirm and create to finish creating your web ACL.

Web ACL name Americold_UI_Dev CloudWatch metric name AmericoldUIDev

Rules and actions Edit

AWS WAF inspects each web request that an AWS resource receives and compares the request with the conditions in the following rules in the order listed. If a request doesn't match all of the conditions in at least one rule, AWS WAF takes the default action.

If a request matches a condition in a rule, take the corresponding action

Order	Rule	Action
1	Americold_IP_Rule	Block
2	Americold_SQL_Injection_Rule	Block
3	Americold_Size_Constraint_Rule	Block
4	Americold_String_Match_Rule	Block
5	Americold_XSS_Rule	Block
6	Americold_Backend_Rule	Block

If a request doesn't match any rules, take the default action

Default action Allow

AWS resources using this web ACL Edit

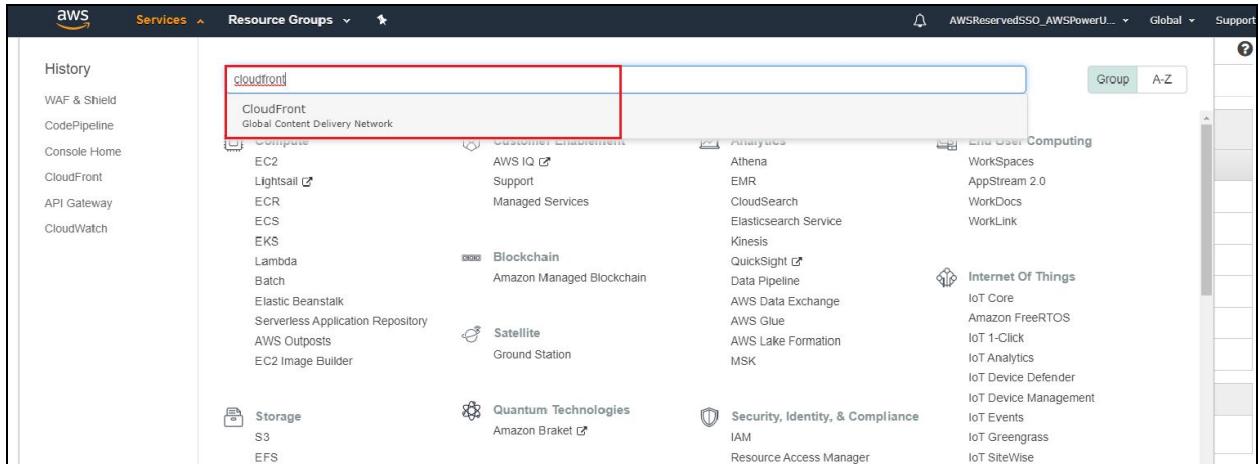
Resource	Type
E3GILM5AHJJTX5 - testinventoryreview.i-3pl.com	CloudFront distribution

Cancel Previous **Confirm and create**

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

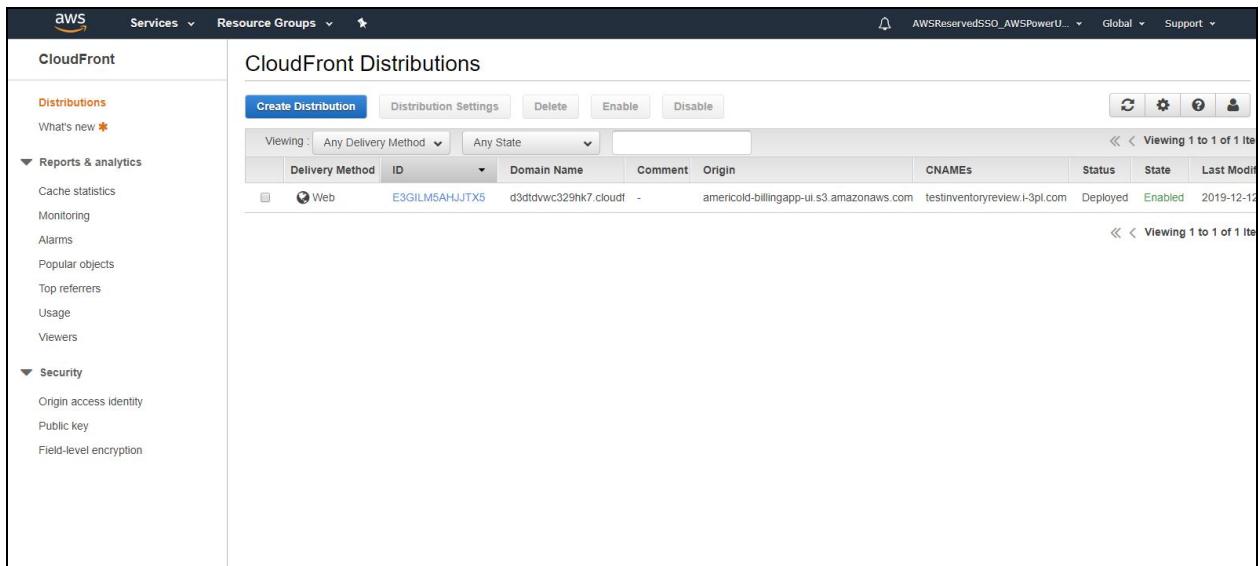
After adding rules to Web ACLs we need to go to AWS Cloudfront service to add our WAF rules to frontend application.

Now search for cloudfront service in console



The screenshot shows the AWS Management Console Services page. A red box highlights the search bar at the top, which contains the text "cloudfront". Below the search bar, the "CloudFront" service is listed in the search results, described as "Global Content Delivery Network". The left sidebar lists various AWS services like WAF & Shield, CloudFront, API Gateway, and CloudWatch. The main pane displays a grid of services under categories such as Compute, Customer Engagement, Analytics, End User Computing, Internet Of Things, and Security, Identity, & Compliance.

Now the dashboard will be as follows



The screenshot shows the AWS CloudFront Distributions dashboard. On the left, there is a navigation sidebar with sections for "Distributions", "Reports & analytics", and "Security". The main area is titled "CloudFront Distributions" and features a "Create Distribution" button. Below it is a search bar and a table listing a single distribution. The table columns include Delivery Method, ID, Domain Name, Comment, Origin, CNAMEs, Status, State, and Last Modified. The distribution listed is "Web" with ID "E3GILM5AHJJTX5", domain name "d3tdtvwc329hk7.cloudf...", origin "americold-billingapp-s3.amazonaws.com", and status "Deployed".

Now we are adding the WAF rules to already created Cloudfront distribution.

The screenshot shows the AWS CloudFront Distributions page. On the left, there's a sidebar with 'CloudFront' selected. Under 'Distributions', it says 'What's new *'. Below that is a section for 'Reports & analytics' with links like Cache statistics, Monitoring, Alarms, Popular objects, Top referrers, Usage, and Viewers. There's also a 'Security' section with links for Origin access identity, Public key, and Field-level encryption. The main area is titled 'CloudFront Distributions' and contains a table with one row. The row has columns for Delivery Method (Web), ID (E3GILM5AHJJTX5), Domain Name (d3ddwvc329hk7.cloudfr...), Comment (-), Origin (americold-billingapp-ui.s3.amazonaws.com), CNAMEs (testinventoryreview.i-3pl.com), Status (Deployed), State (Enabled), and Last Modified (2019-12-12). A red box highlights this row. At the top of the main area, there are buttons for 'Create Distribution', 'Distribution Settings', 'Delete', 'Enable', and 'Disable'. The status bar at the bottom right says 'Viewing 1 to 1 of 1 item'.

Now select the Cloudfront distribution and click on **edit**

The screenshot shows the 'General' tab of a CloudFront distribution's configuration page. The distribution ID is E3GILM5AHJJTX5. The 'Edit' button is highlighted with a red box. The page lists various settings: ARN (arn:aws:cloudfront::646632670602:distribution/E3GILM5AHJJTX5), Log Prefix (-), Delivery Method (Web), Cookie Logging (Off), Distribution Status (Deployed), Comment (-), Price Class (Use All Edge Locations (Best Performance)), AWS WAF Web ACL (Americold_UI_Dev), and State (Enabled). Under 'Alternate Domain Names (CNAMEs)', it shows testinventoryreview.i-3pl.com. It also lists SSL Certificate (.i-3pl.com (9d59cfed-eca3-43c8-8ede-6ce39b6e7fae)), Domain Name (d3ddwvc329hk7.cloudfront.net), Custom SSL Client Support (Clients that Support Server Name Indication (SNI) - (Recommended)), Security Policy (TLSv1_2_2016), Supported HTTP Versions (HTTP/2, HTTP/1.1, HTTP/1.0), IP Version (IPV6 Enabled), Default Root Object (index.html), Last Modified (2019-12-12 19:12 UTC+5:30), and Log Bucket (americold.log.ui.s3.amazonaws.com). The left sidebar is identical to the previous screenshot.

After click on the edit the dashboard is shown as below

The screenshot shows the 'Edit Distribution' page for a CloudFront distribution. Under 'Distribution Settings', the 'AWS WAF Web ACL' dropdown is set to 'None'. The 'Alternate Domain Names (CNAMEs)' field contains 'testinventoryreview.i-3pl.com'. The 'SSL Certificate' section shows 'Default CloudFront Certificate (*.cloudfront.net)' selected. The 'Custom SSL Client Support' section shows 'Clients that Support Server Name Indication (SNI) - (Recommended)' selected.

Now click on the drop down that was showing for **AWS WAF Web ACL** and select the Web ACL we have created in the above steps

The screenshot shows the same 'Edit Distribution' page as before, but the 'AWS WAF Web ACL' dropdown has been interacted with. A red box highlights the dropdown menu, which now shows 'None' and 'American_Web_ACL' as options. The 'Alternate Domain Names (CNAMEs)' field still contains 'testinventoryreview.i-3pl.com'. The 'SSL Certificate' and 'Custom SSL Client Support' sections remain the same.

Now click on the Yes, Edit in the bottom of the page to update the changes.

The screenshot shows the AWS CloudFront distribution configuration page. The 'General' tab is selected. Key settings include:

- Security Policy:** TLSv1_2016 (recommended)
- Supported HTTP Versions:** HTTP/2, HTTP/1.1, HTTP/1.0
- Default Root Object:** Index.html
- Logging:** On
- Bucket for Logs:** americold.log.ul.s3.amazonaws.com
- Log Prefix:** (empty)
- Cookie Logging:** Off
- Enable IPv6:** checked
- Comment:** (empty)
- Distribution State:** Enabled

At the bottom right, there are 'Cancel' and 'Yes, Edit' buttons, with 'Yes, Edit' being highlighted.

Now we can see WAF has been added to the Cloudfront distribution

The screenshot shows the AWS CloudFront distribution details page for distribution ID E3GILM5AHJJTX5. The 'General' tab is selected. Key configuration details include:

- Distribution ID:** E3GILM5AHJJTX5
- ARN:** arn:aws:cloudfront:646632670602:distribution/E3GILM5AHJJTX5
- Log Prefix:** -
- Delivery Method:** Web
- Cookie Logging:** Off
- Distribution Status:** Deployed
- Comment:** -
- Price Class:** Use All Edge Locations (Best Performance)
- AWS WAF Web ACL:** Americold_UI_Dev (highlighted)
- State:** Enabled
- Alternate Domain Names (CNAMEs):** testinventoryreview.i-3pl.com, *-i-3pl.com (9d59cfed-eca3-43c8-8ede-6ce39b6e7fae)
- SSL Certificate:** *-i-3pl.com (9d59cfed-eca3-43c8-8ede-6ce39b6e7fae)
- Domain Name:** d3ddtwc329hk7.cloudfront.net
- Custom SSL Client Support:** Clients that Support Server Name Indication (SNI) - (Recommended)
- Security Policy:** TLSv1_2016
- Supported HTTP Versions:** HTTP/2, HTTP/1.1, HTTP/1.0
- IPv6:** Enabled
- Default Root Object:** index.html
- Last Modified:** 2019-12-12 19:12 UTC+5:30
- Log Bucket:** americold.log.ul.s3.amazonaws.com

In the similar way we have added WAF for QA, Stage (UAT) and Prod environments.

----- End of the Document -----