WAF for Frontend

WAF Introduction:

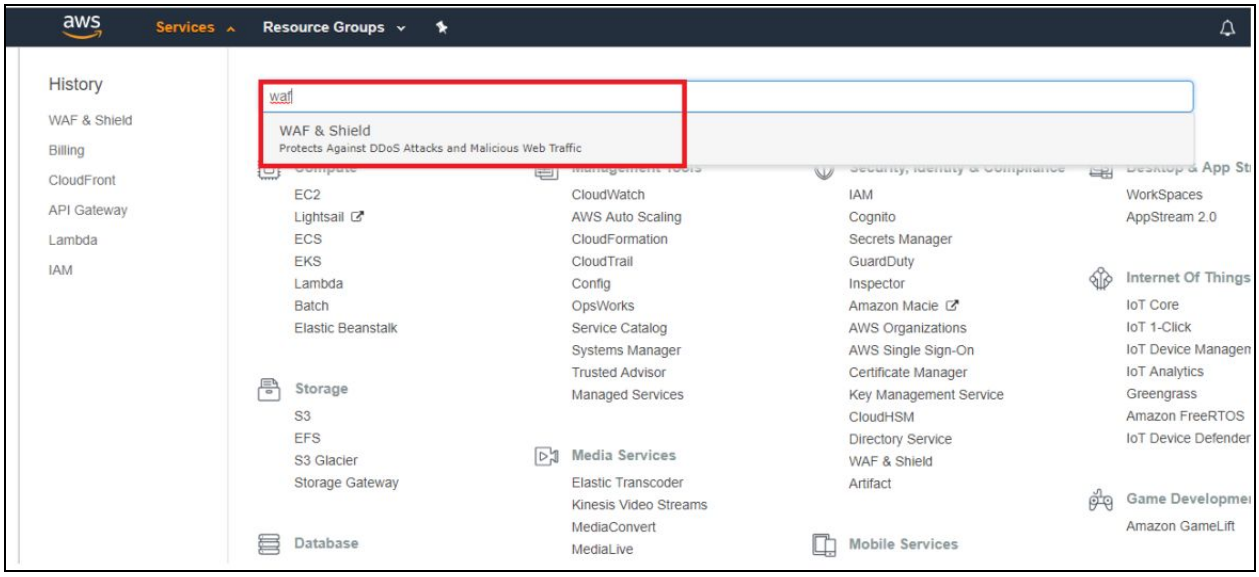
AWS **WAF** is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources.

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.

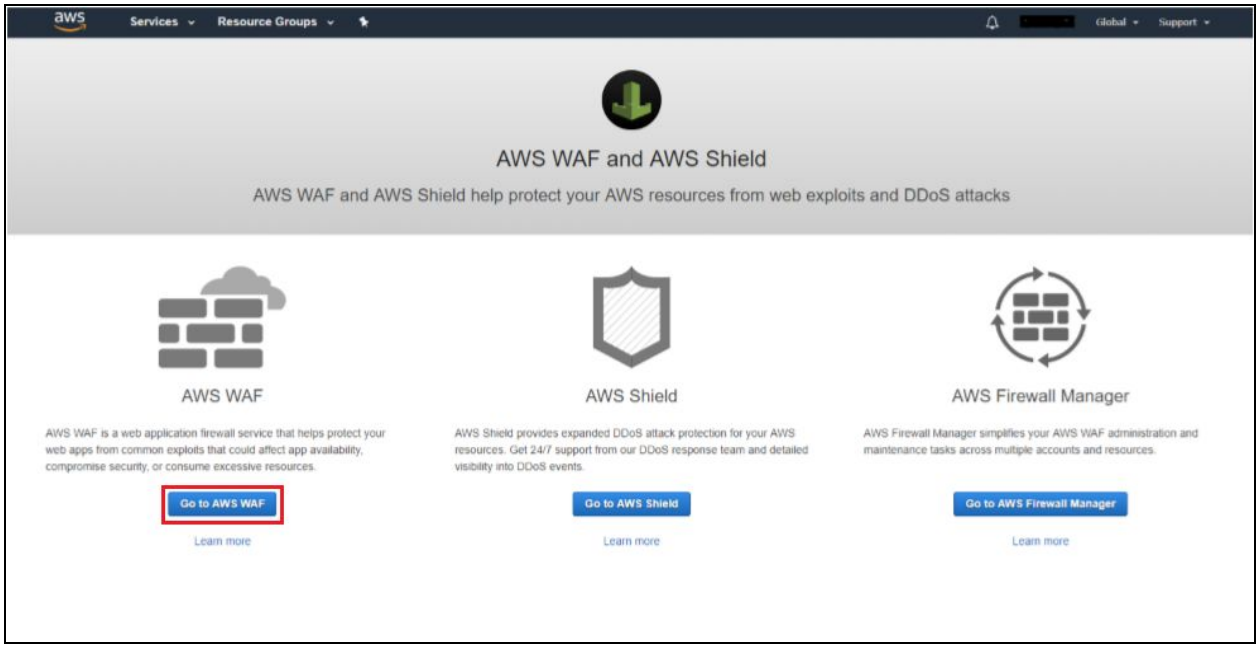
With AWS WAF you pay only for what you use. AWS WAF pricing is based on how many rules you deploy and how many web requests your web application receives. There are no upfront commitments.

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.

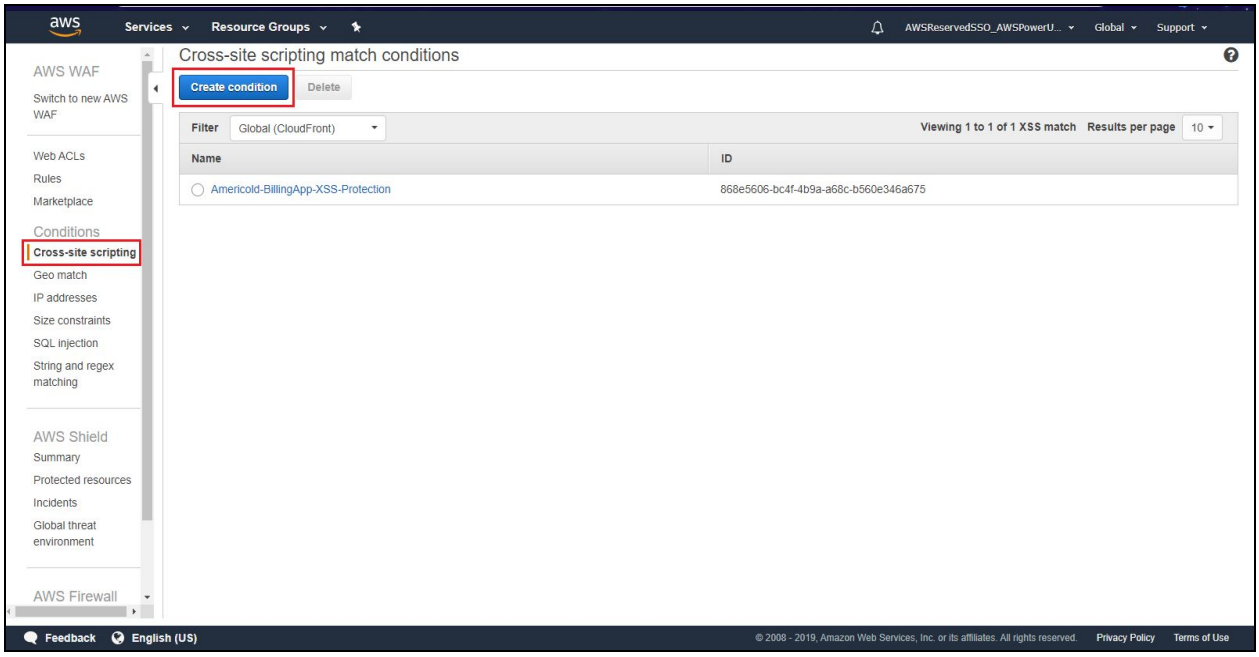
First, we have to login into the **AWS Console** and search for the WAF service.



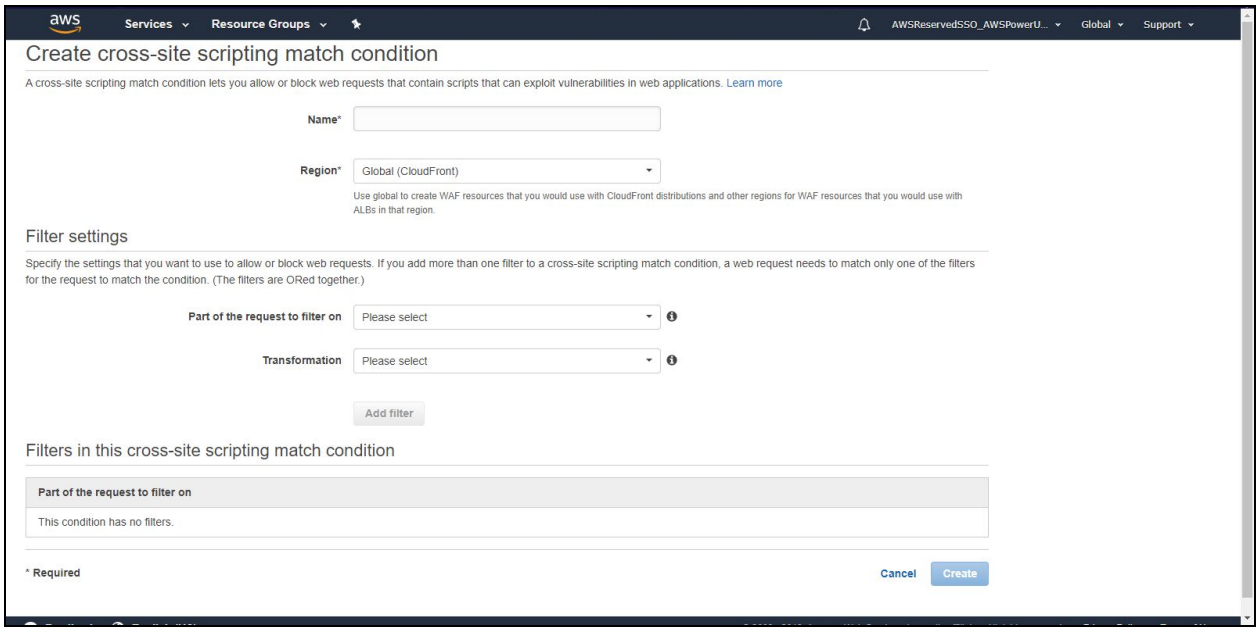
Following is the dashboard for AWS WAF. Now click on goto AWS WAF



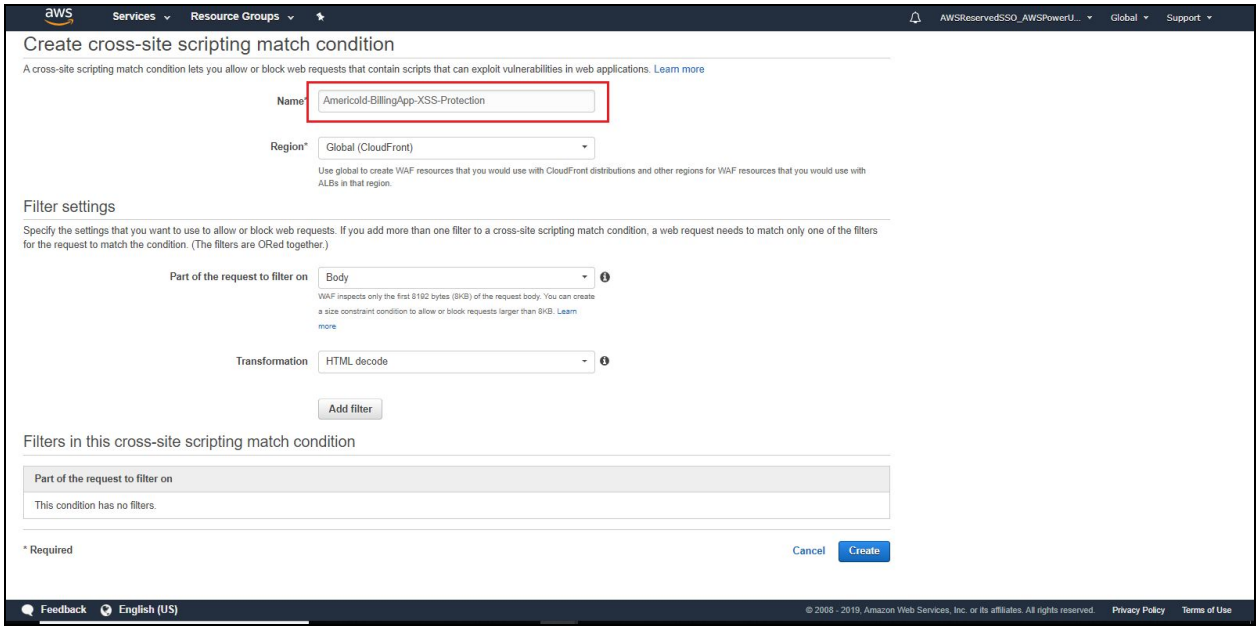
Now we will be creating cross site scripting conditions so we will be selecting **Cross site scripting** and click on **Create Conditions** as shown below.



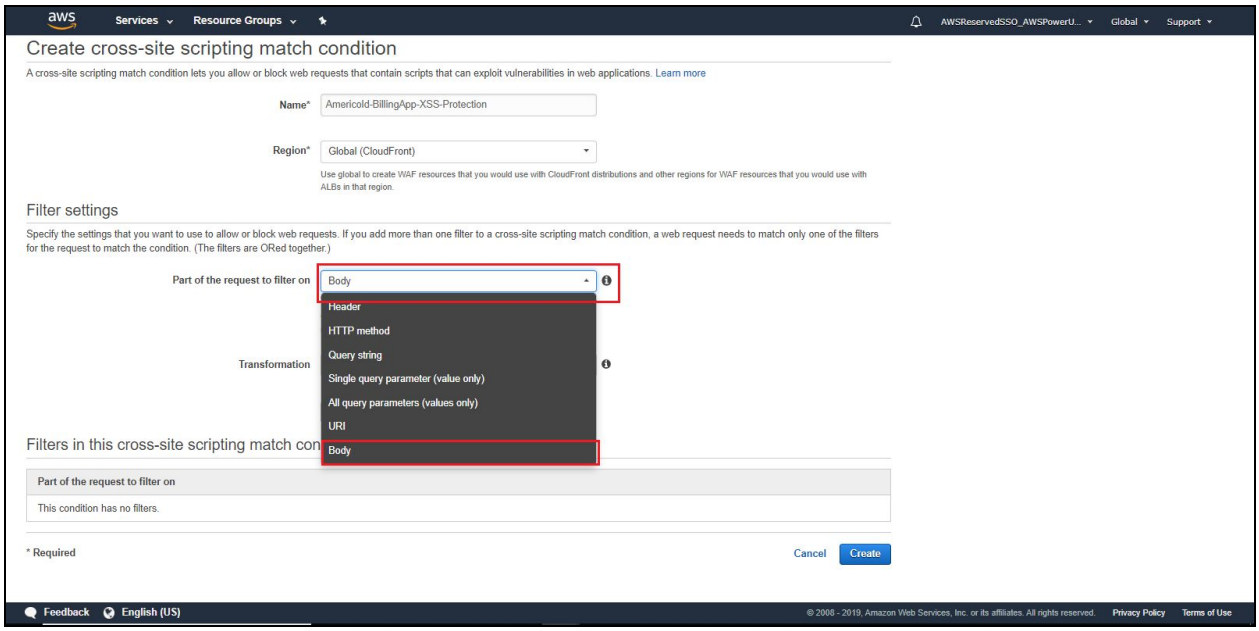
After clicking on create condition we will get the dashboard as follows to add the cross-site scripting conditions.



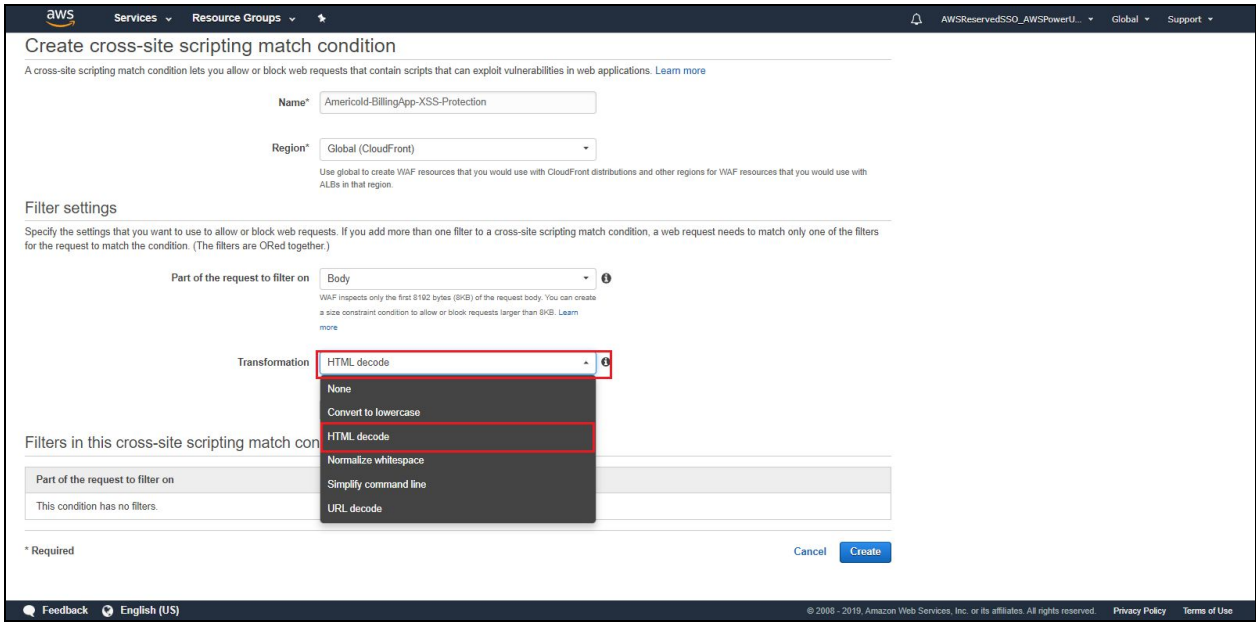
Here we gave the condition name as **Americold-BillingApp-XSS-Protection.**



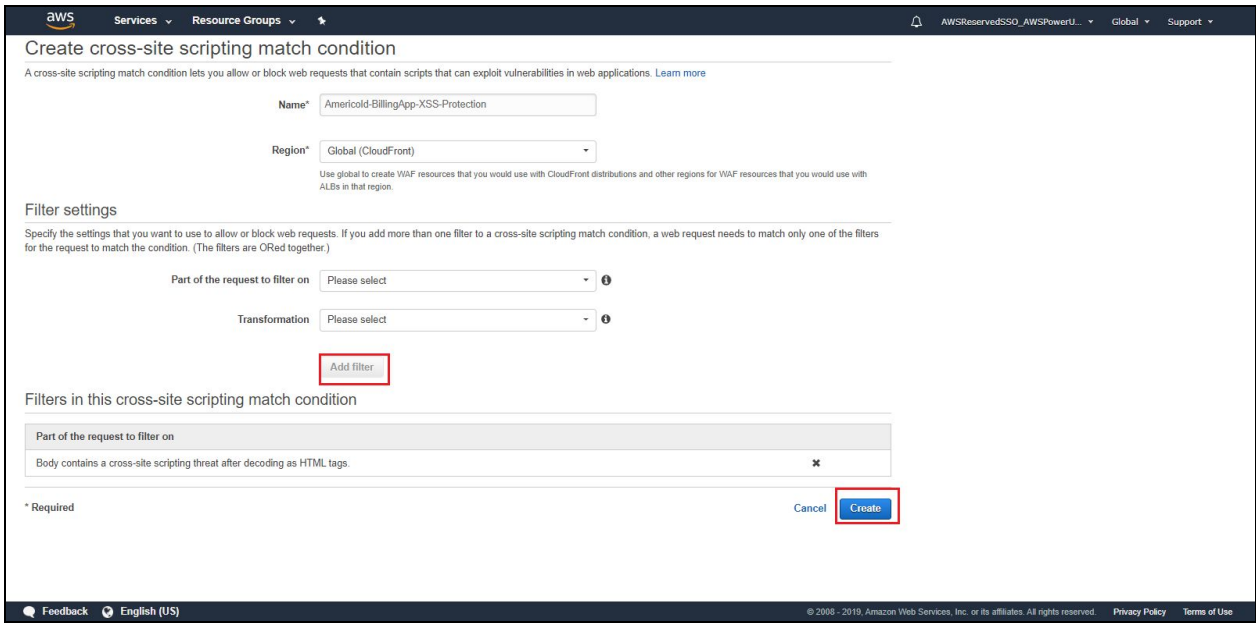
Now select **Body** in **Part of the request to filter** on to specify that where the condition have to apply



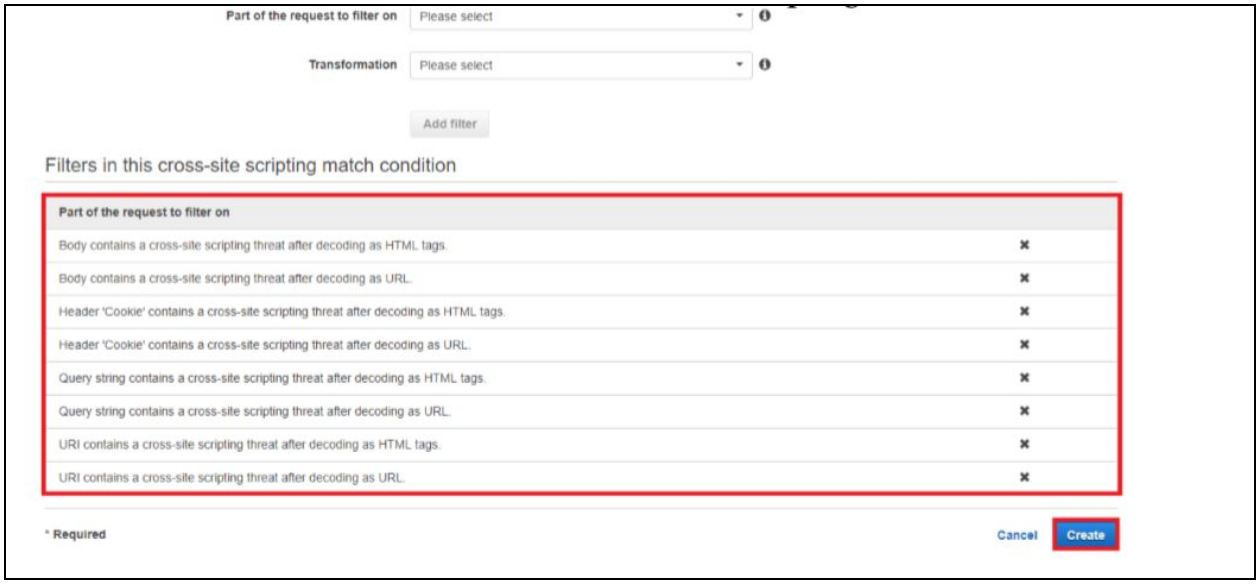
Now select **HTML decode** in **Transformation.**



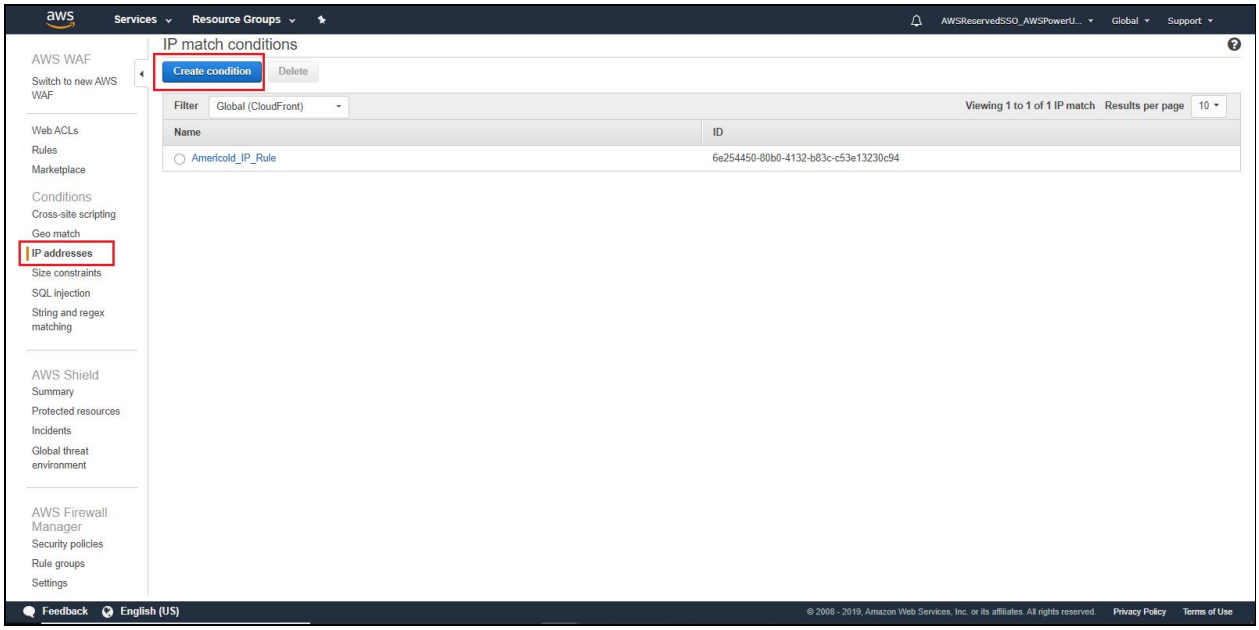
Now click on the **Add filter** to add the filter.



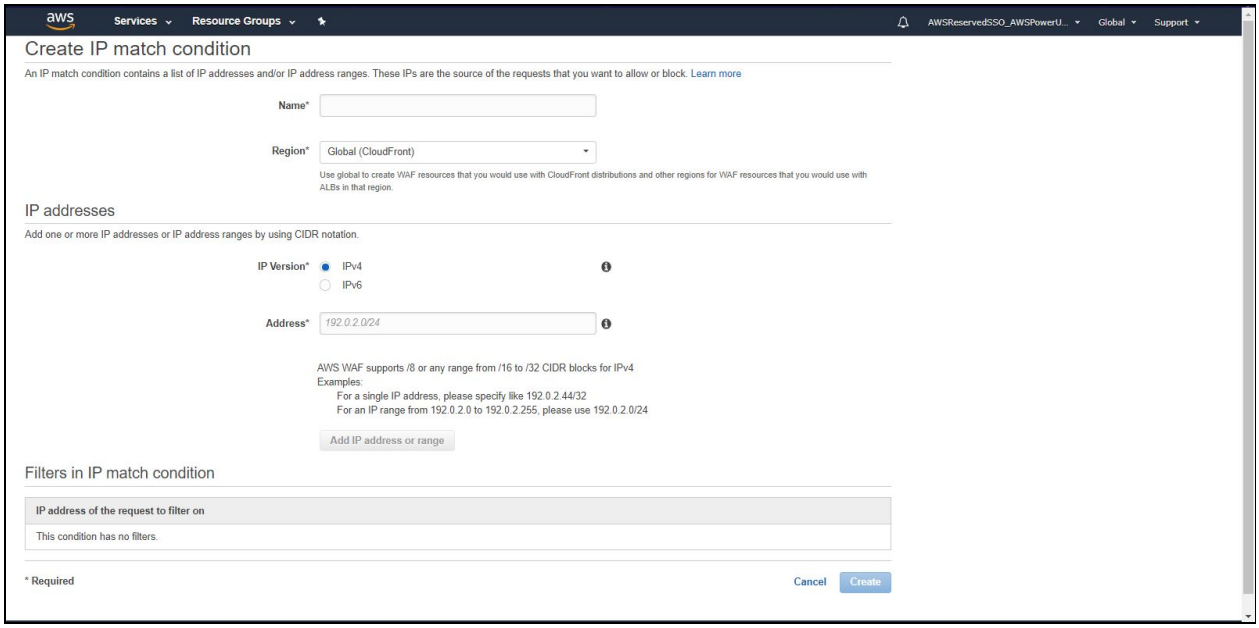
We have to add all the filters that what we need and click on Create to create **Cross site scripting** as shown below.



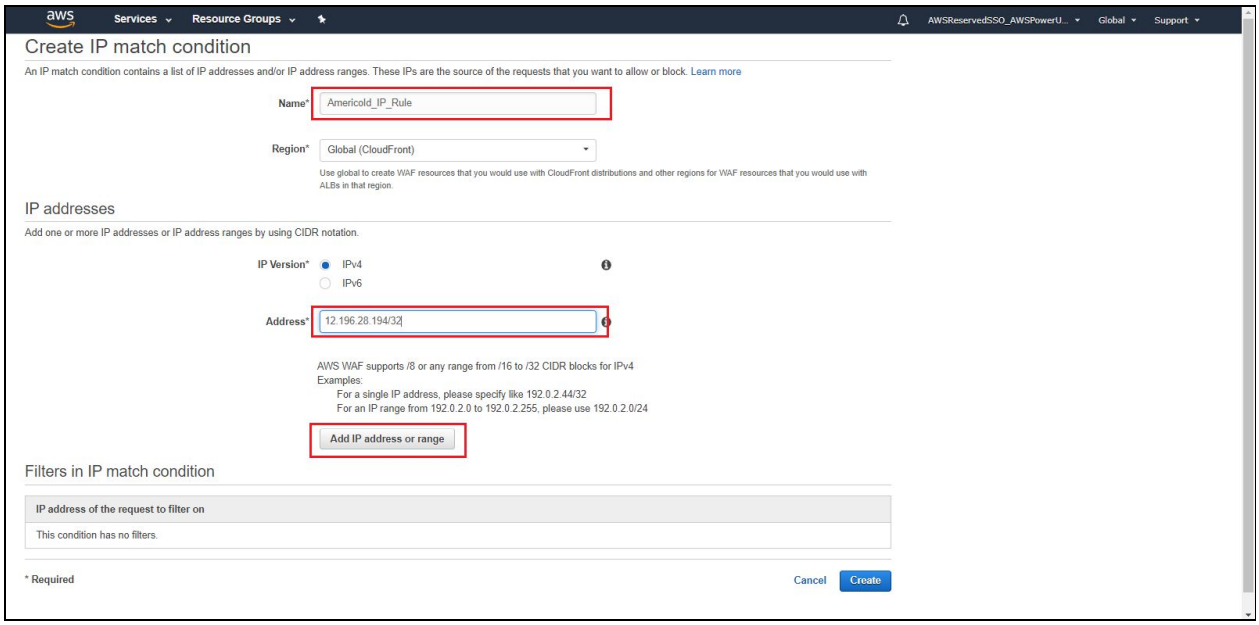
Now go to **IP addresses** section and click on **create condition** to give the access for specific IP addresses. Here we are adding the **publicIP/CIDR** of Americold network which is provided by Americold networking team. It allows us to access the application with in the Americold network.



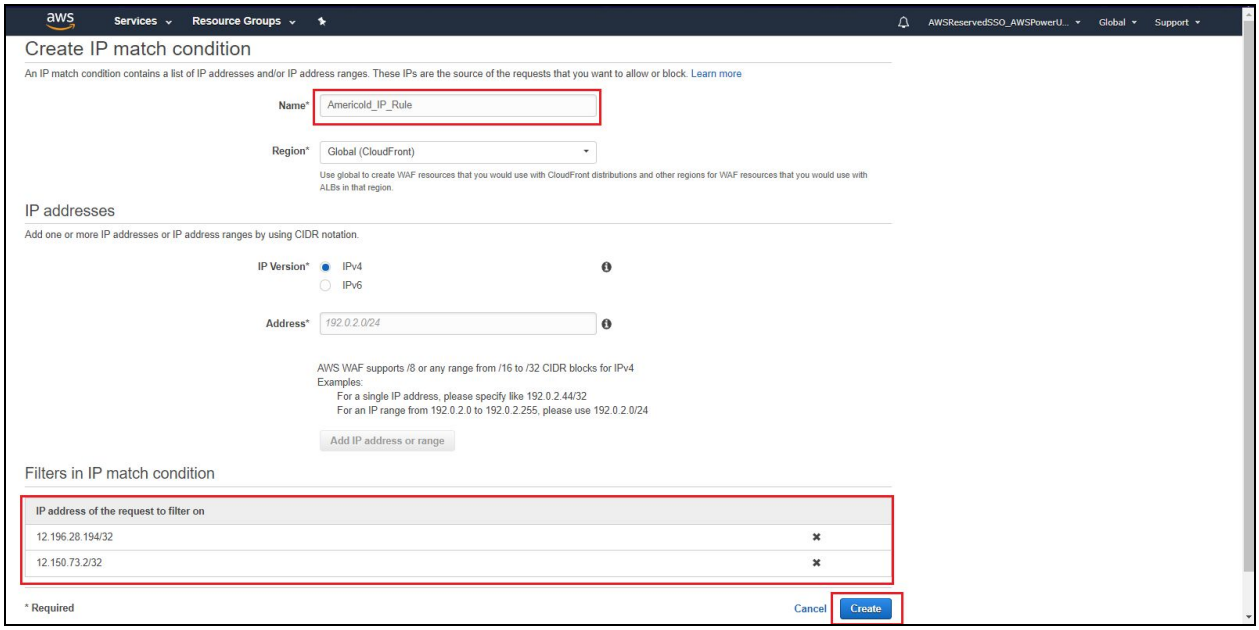
Now we will get the dashboard as follows to configure the IP addresses



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

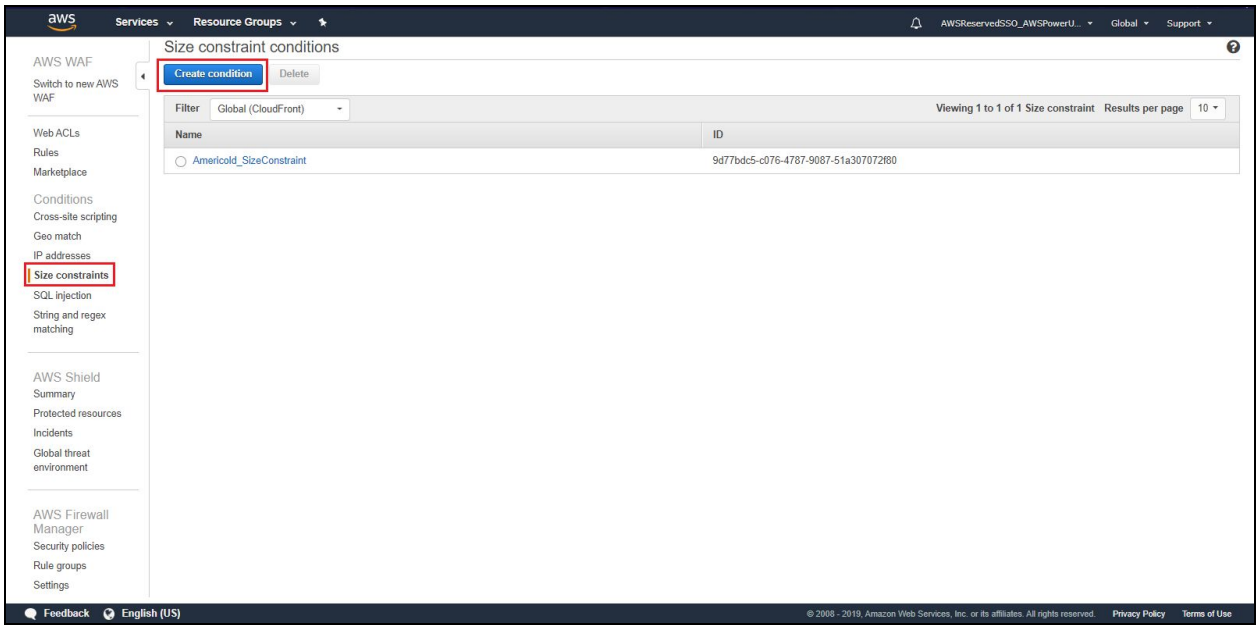


Now the IP address will be added to the **Filter in IP match condition** and then click on **create.** In the same way that we have mentioned above we have to add all IP addresses that can access our application

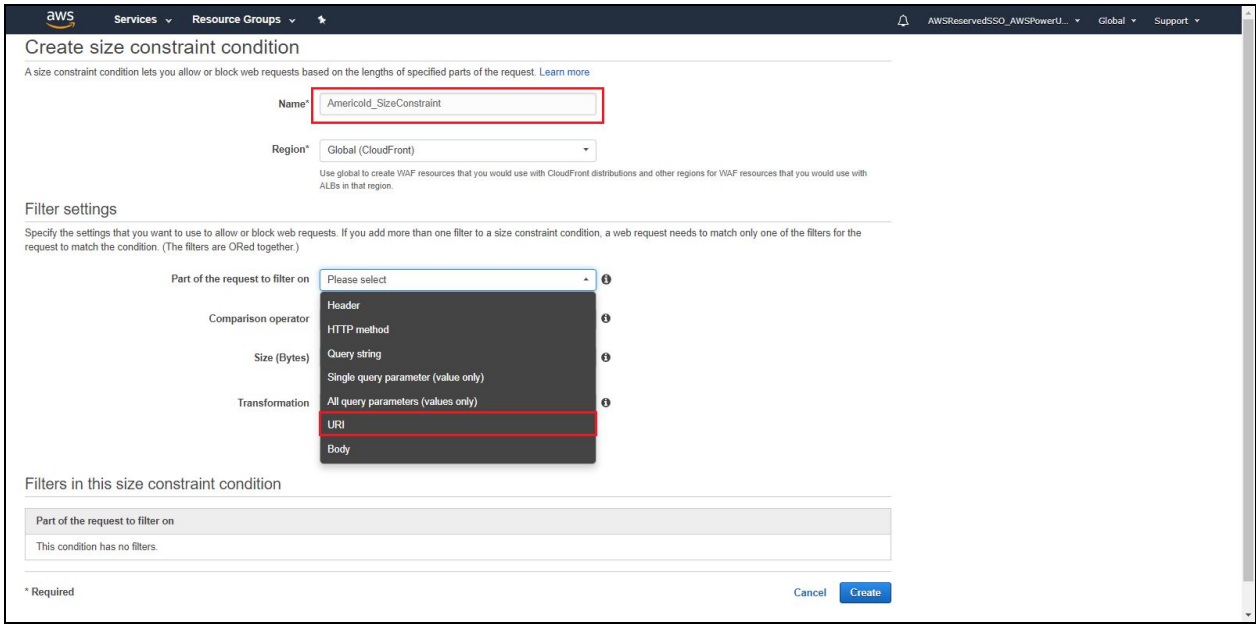


A **size constraint** statement compares a number of bytes against the size of a request component, using a comparison operator, such as greater than (>) or less than (<). If the size of the string not satisfies the condition we can restrict the access.

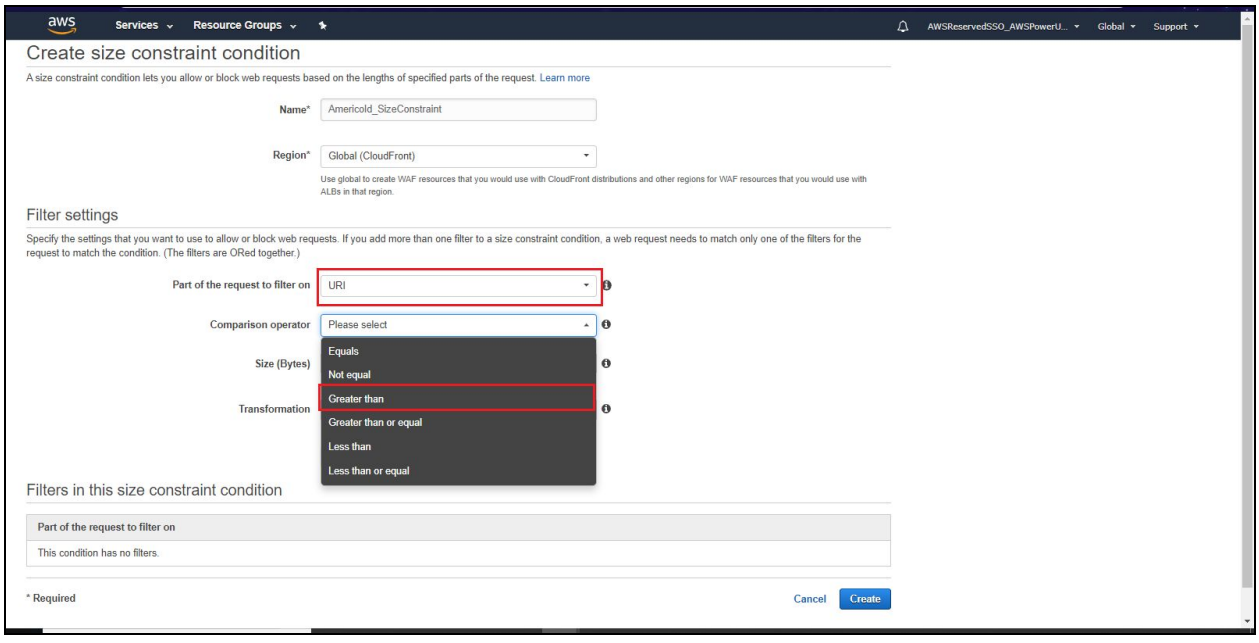
Now go to **Size constraints** and click on **Create Conditions.**



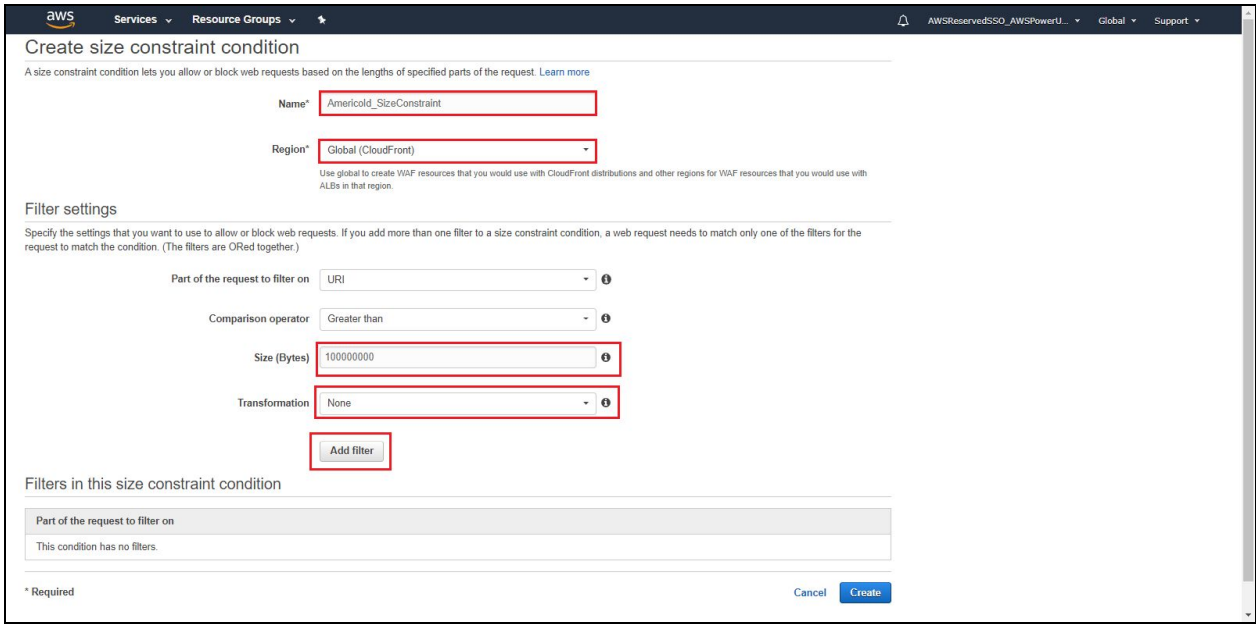
Now select **URI** in **Part of the request to filter on** to specify that where the condition have to apply



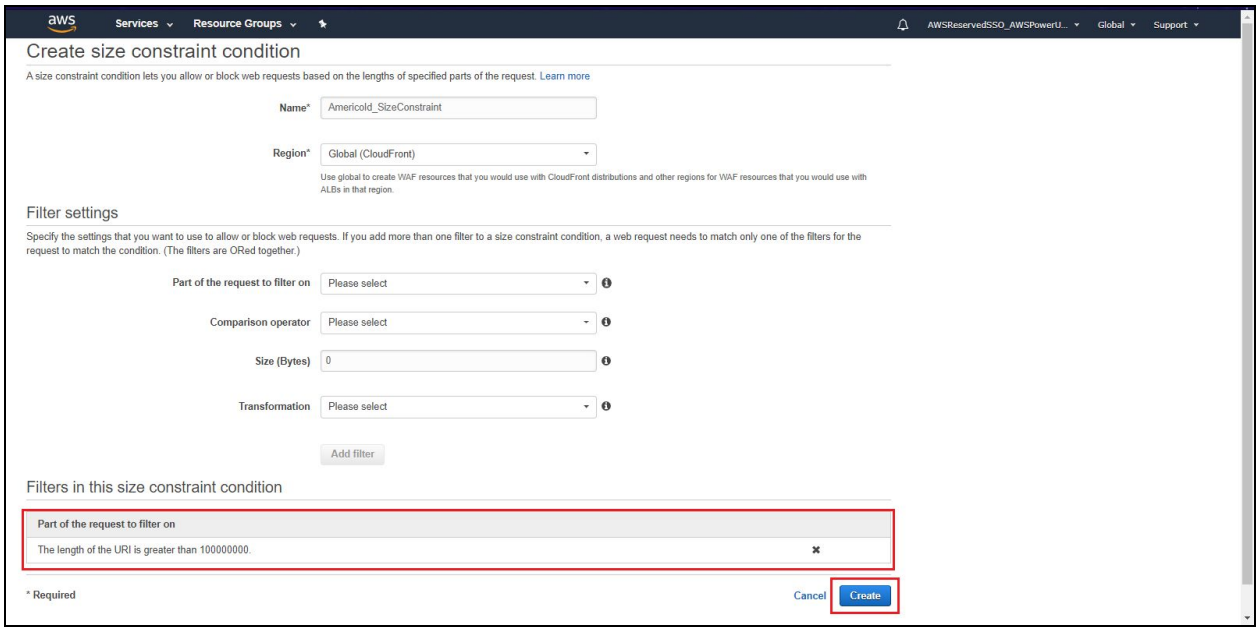
Now select **Greater than** in **Comparison operator**.



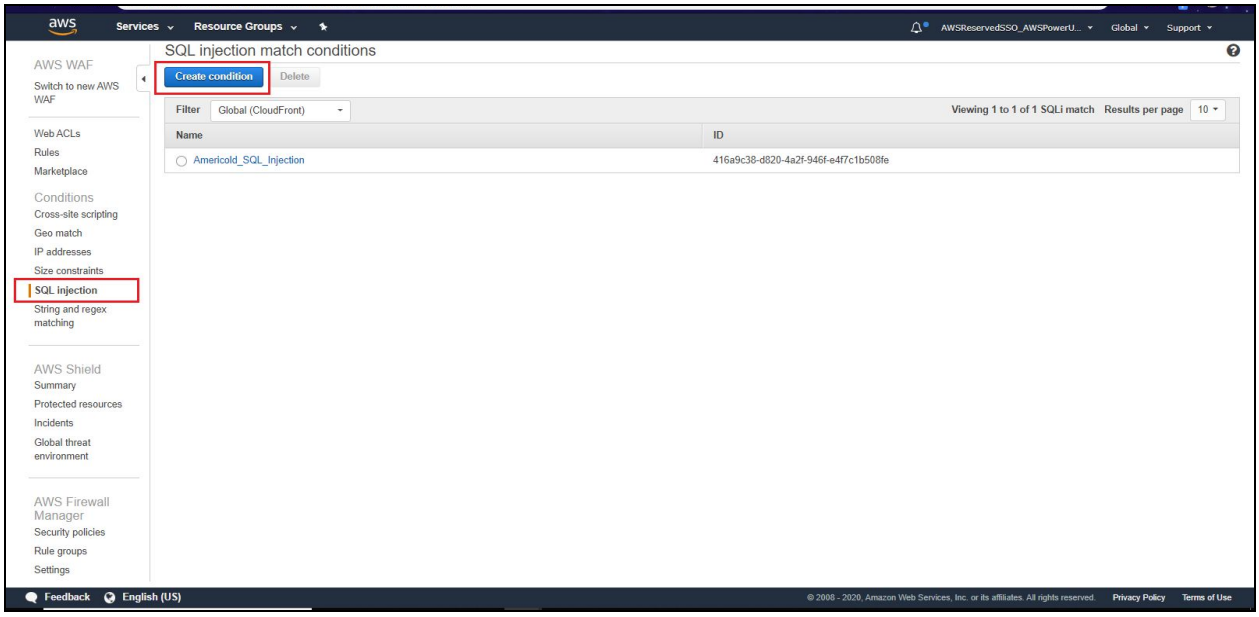
Now we gave the name as **Size Constraint**, Region will be as default, select **URI** in the part of the request to filter on, select **Greater than** in the comparison operator, give the size as **100000000** and select **None** in the Transformation then click on the **Add filter** button.



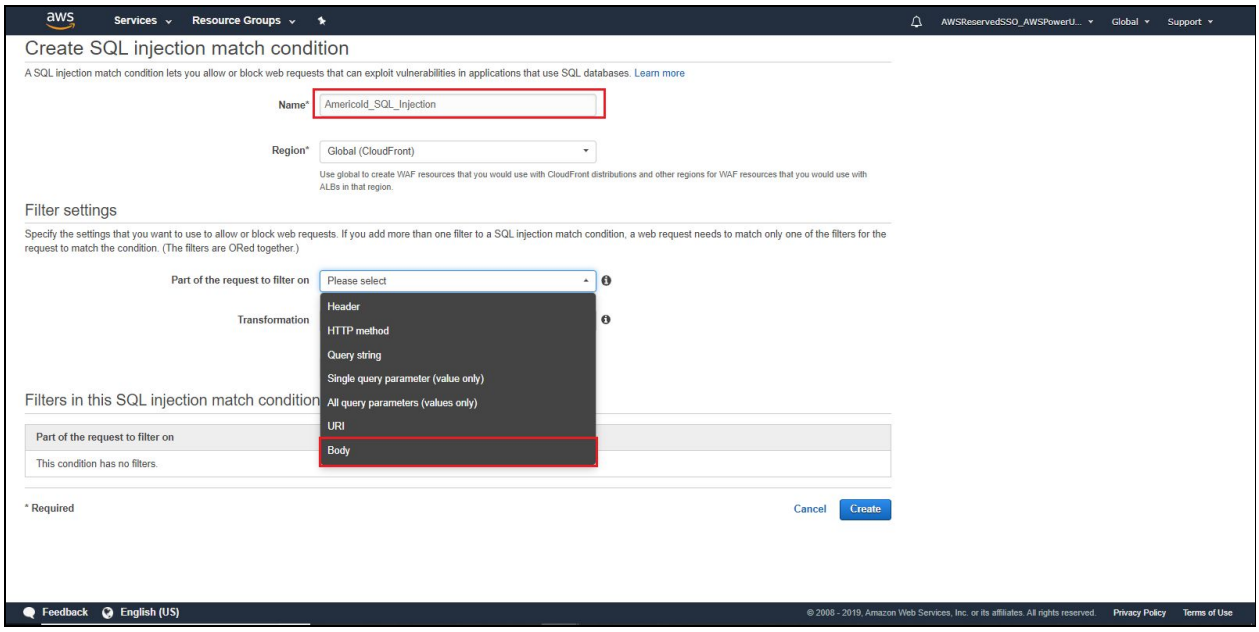
Now the Filter will be added and click on **create** button.



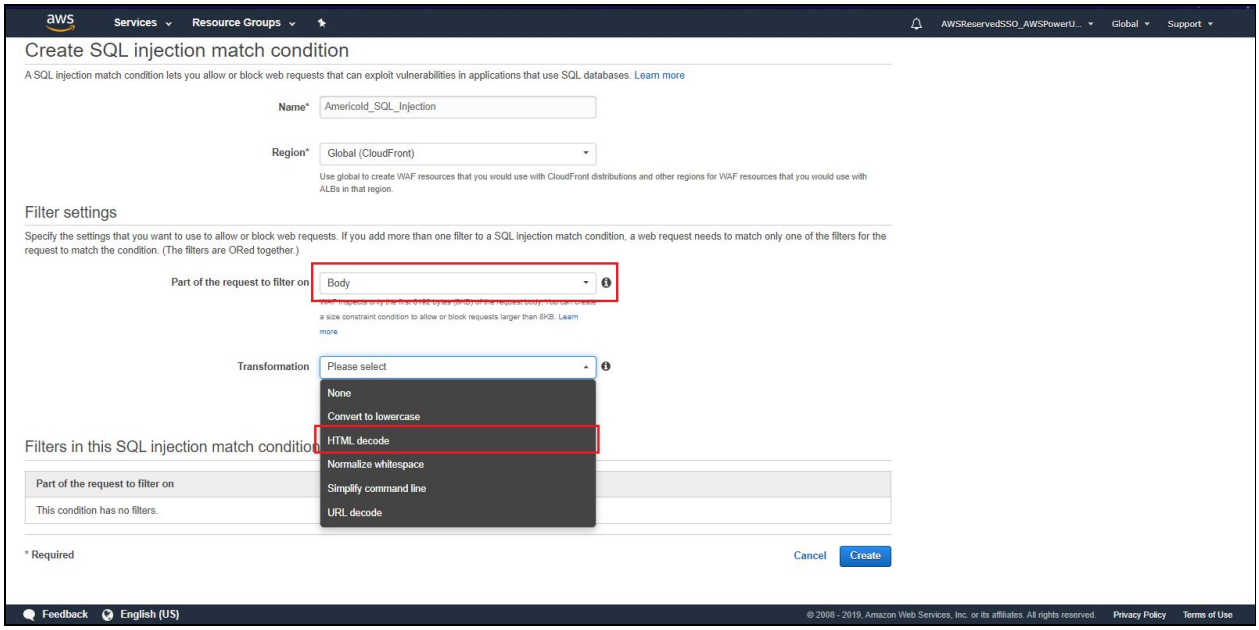
SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. To restrict these types of vulnerabilities we have added SQL injection conditions.
Now go to **SQL Injections** and click on **create conditions** to add the **SQL Injection condition.**



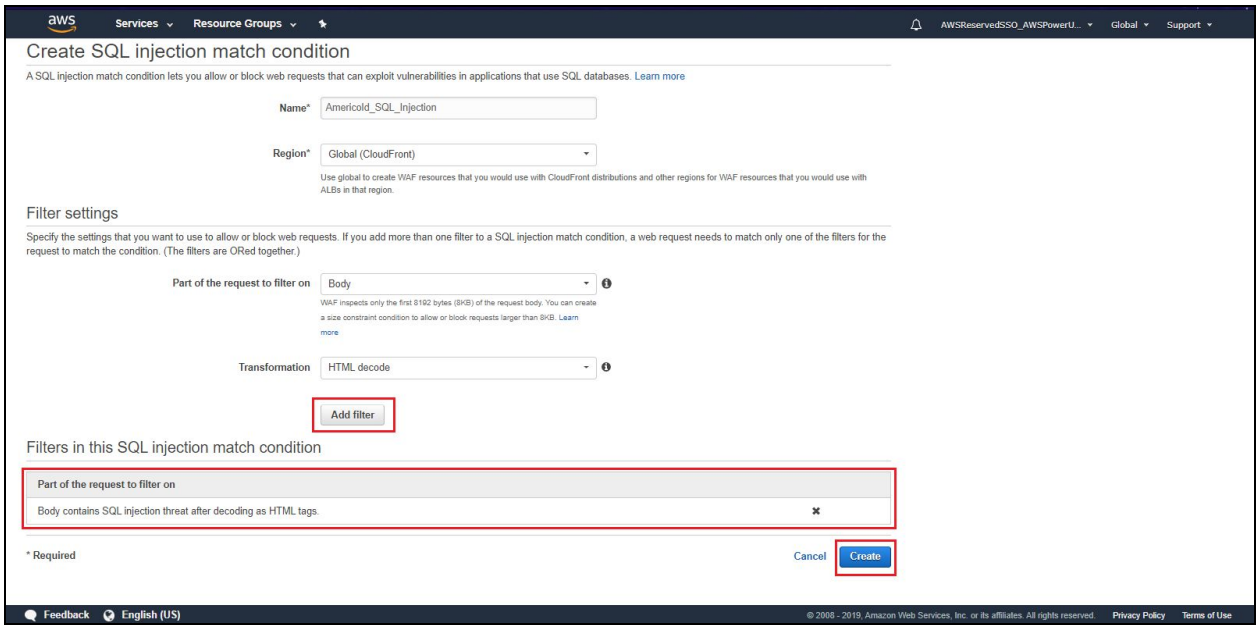
Now select **body** in **Part of the request to filter on** to specify that where the condition have to apply



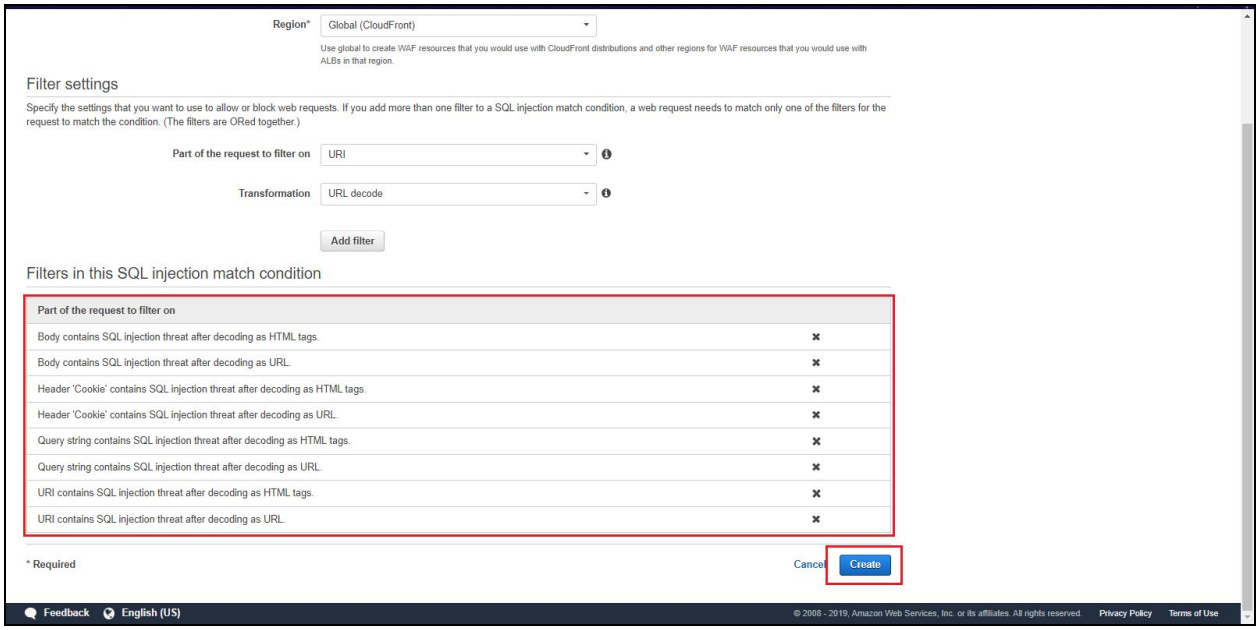
Now select **HTML decode** in **Transformation**.



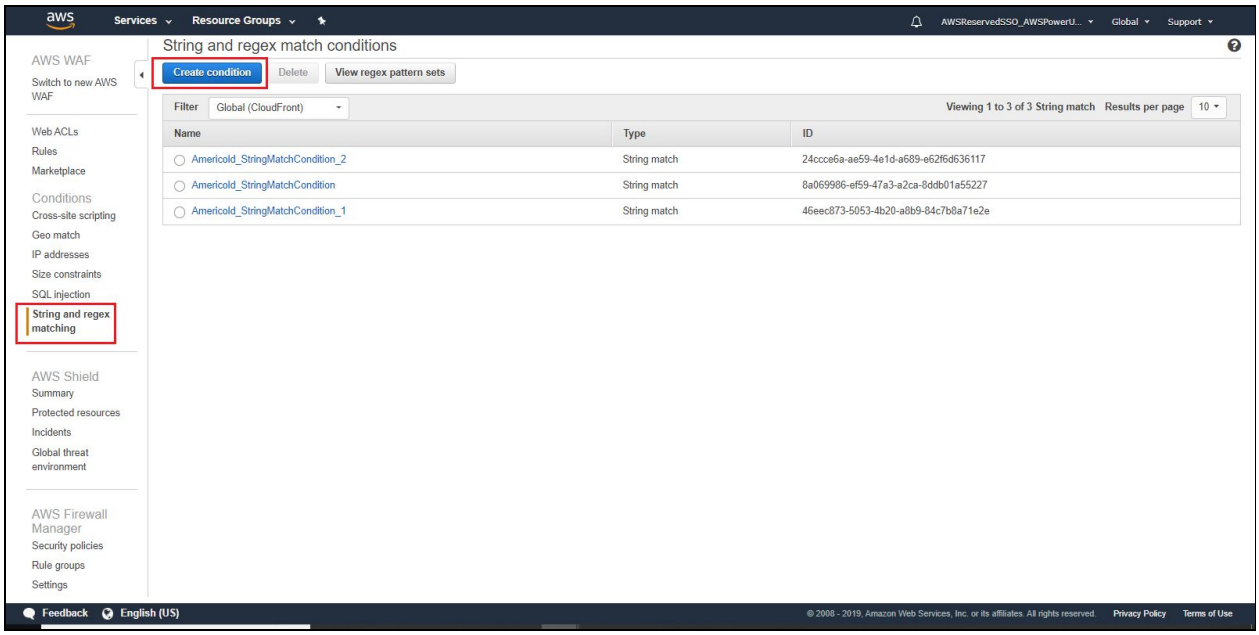
Now the filter will be added as shown below



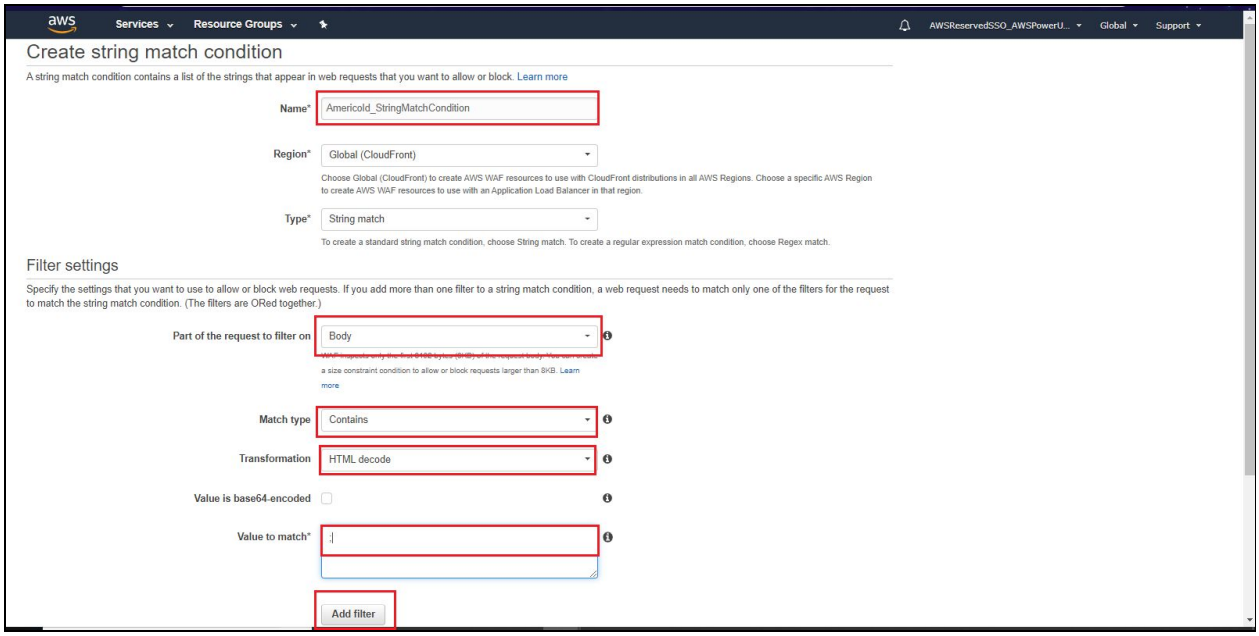
In the same way we will add all filters as per the requirement and click on **create** button to confirm.



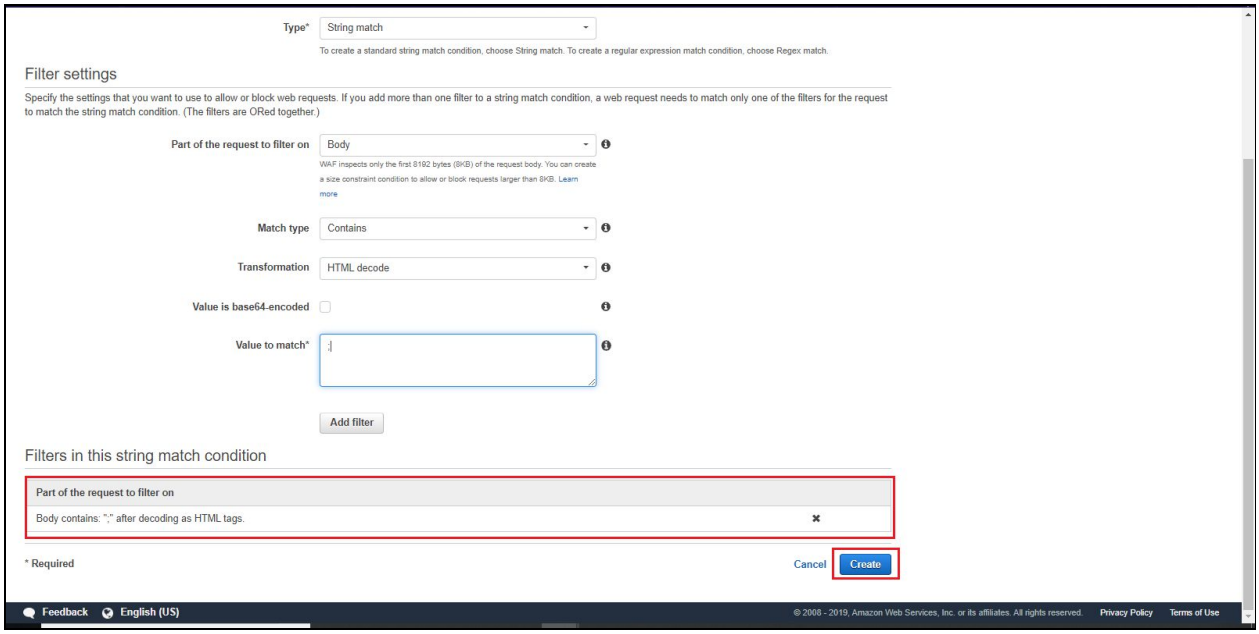
If you want to allow or block web requests based on strings that match a regular expression (regex) pattern that appears in the requests we need to create String and regex match conditions.
Now go to **String and regex matching** and click on **Create condition.**



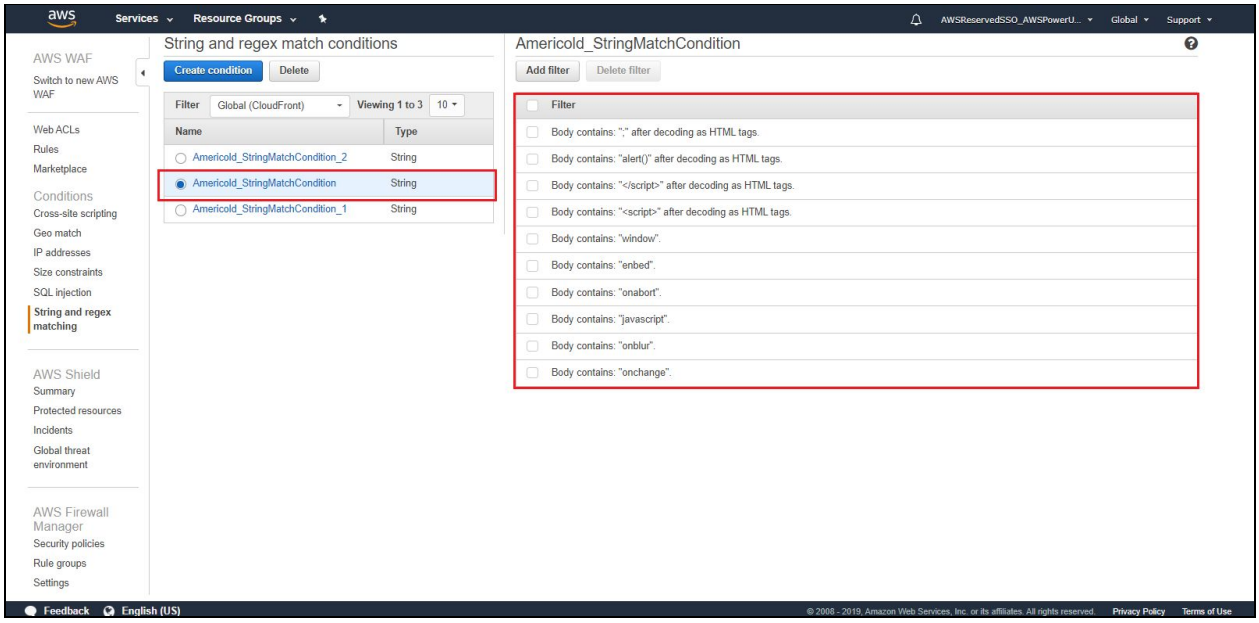
Now configure the name as **BlackList1**, configure the **Type** as **String match** and in **filter settings** section Select **Body** in Part of the request to filter on, **Contains** in match type, select **HTML decode** in the Transformation and some value to match then click on the **Add filter.**



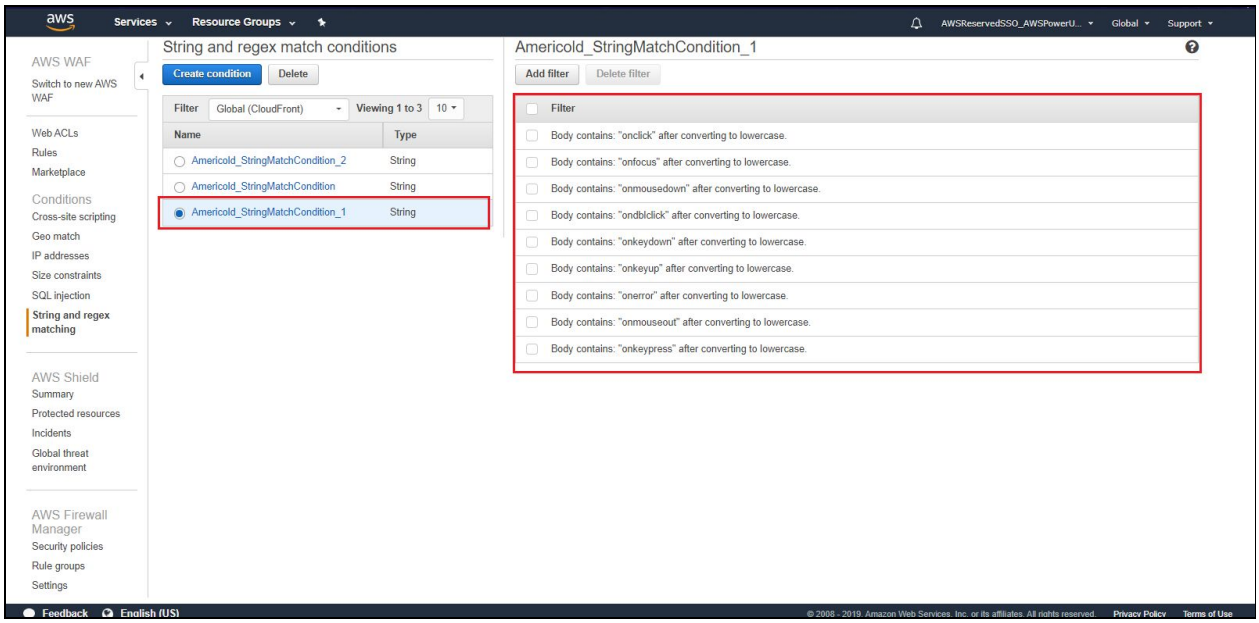
After adding all the filters click on **create** button to confirm.



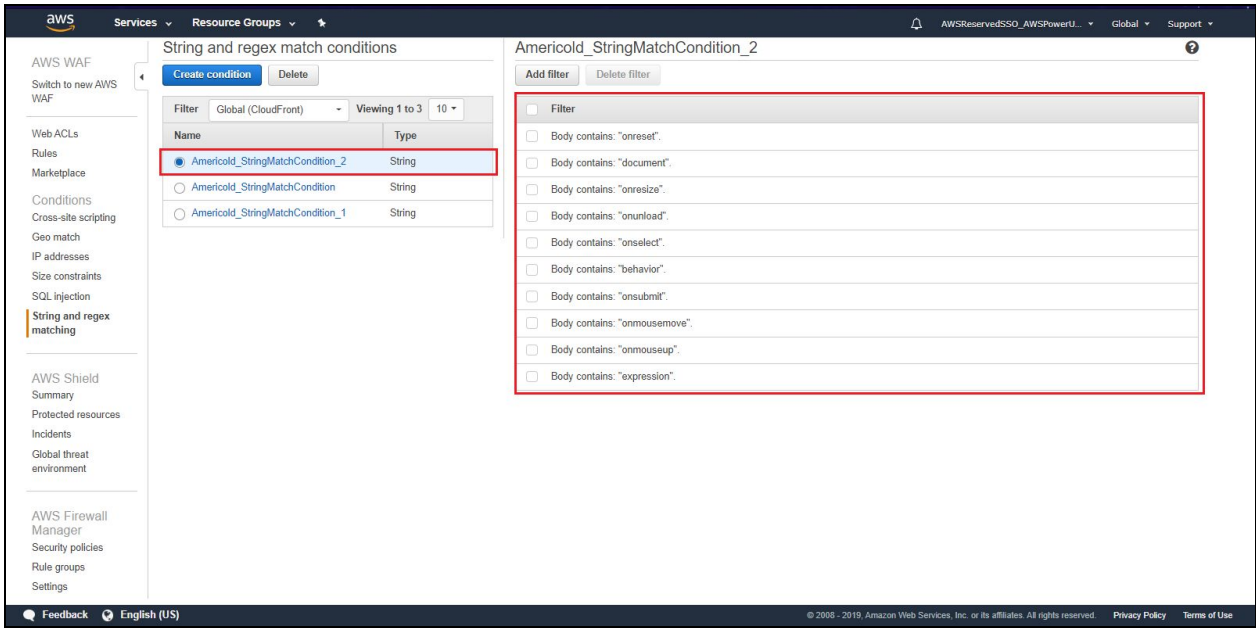
Following is the output for the above steps.



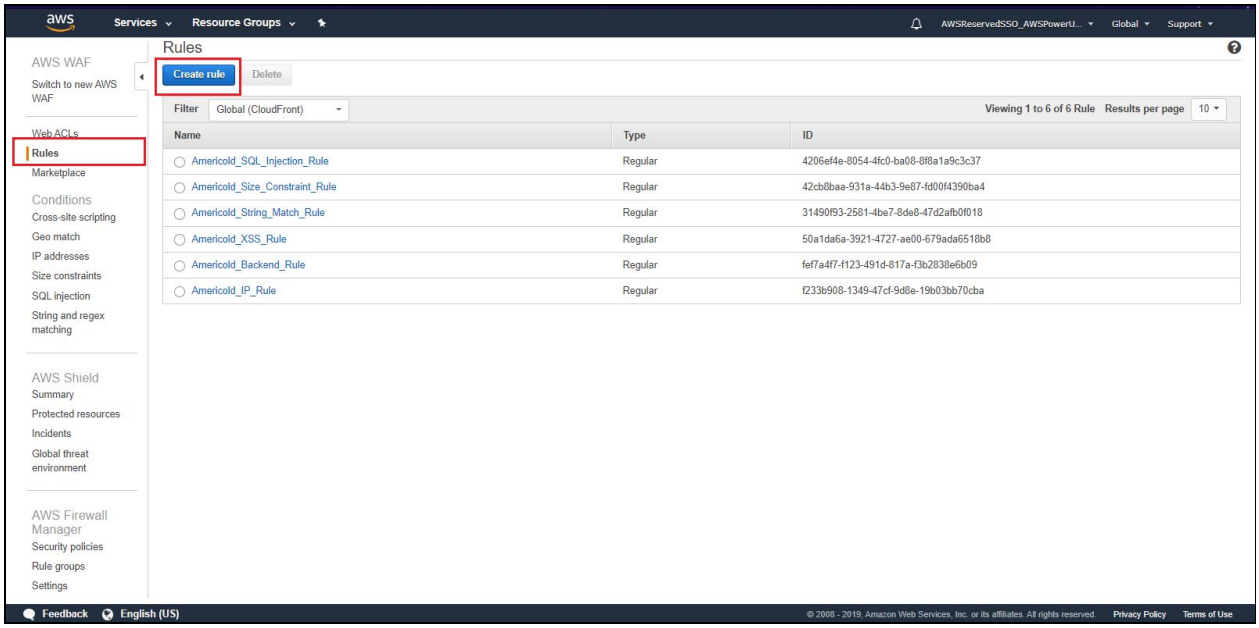
Here we have created another condition in **string and regex matching** as **Americold_StringMatchCondidtion_1**.



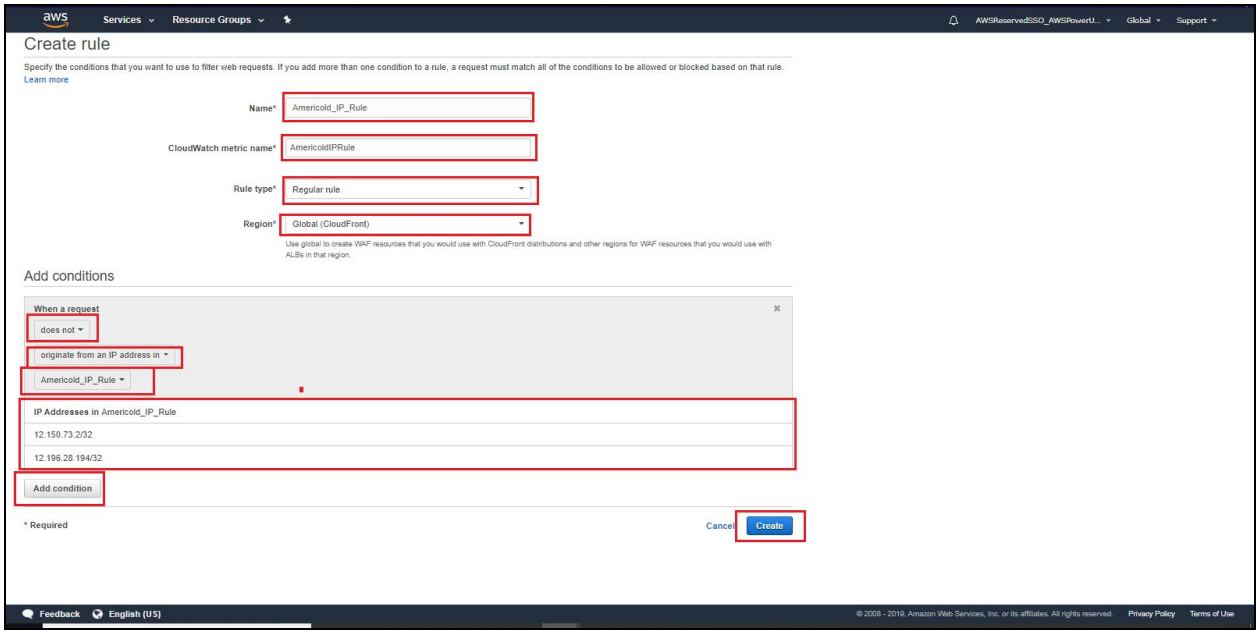
Here we created another condition in string and regex matching as **Americlod_StringMatchCondition_2**.



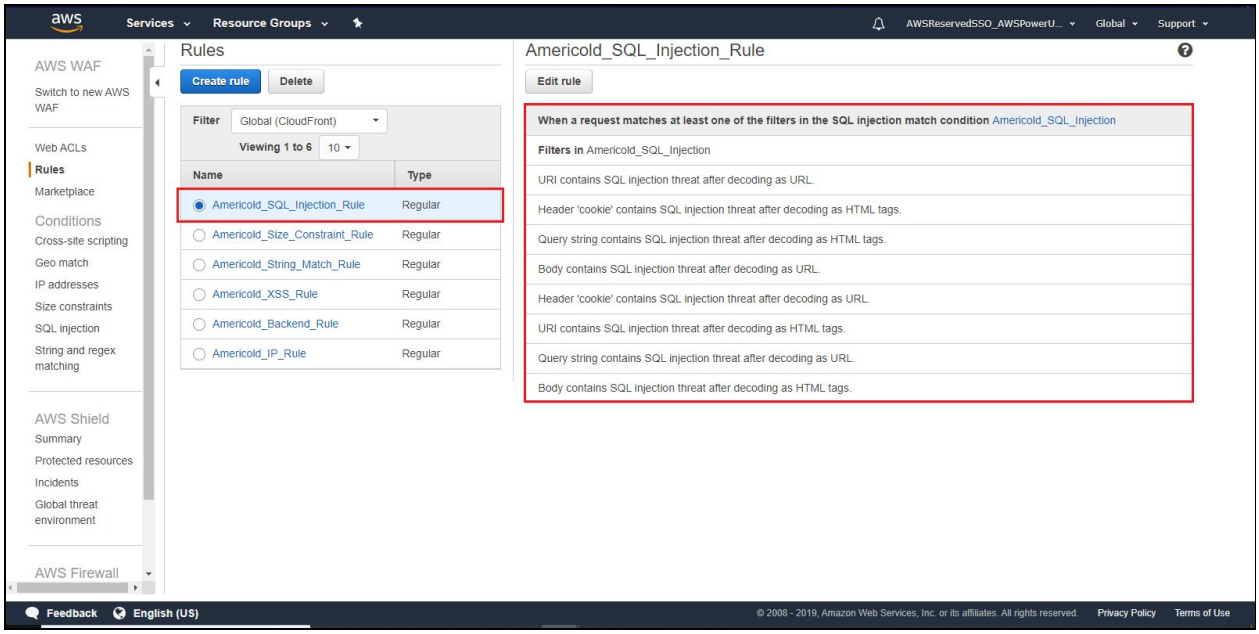
Now go to **Rules section** and click on the **Create Rule.**



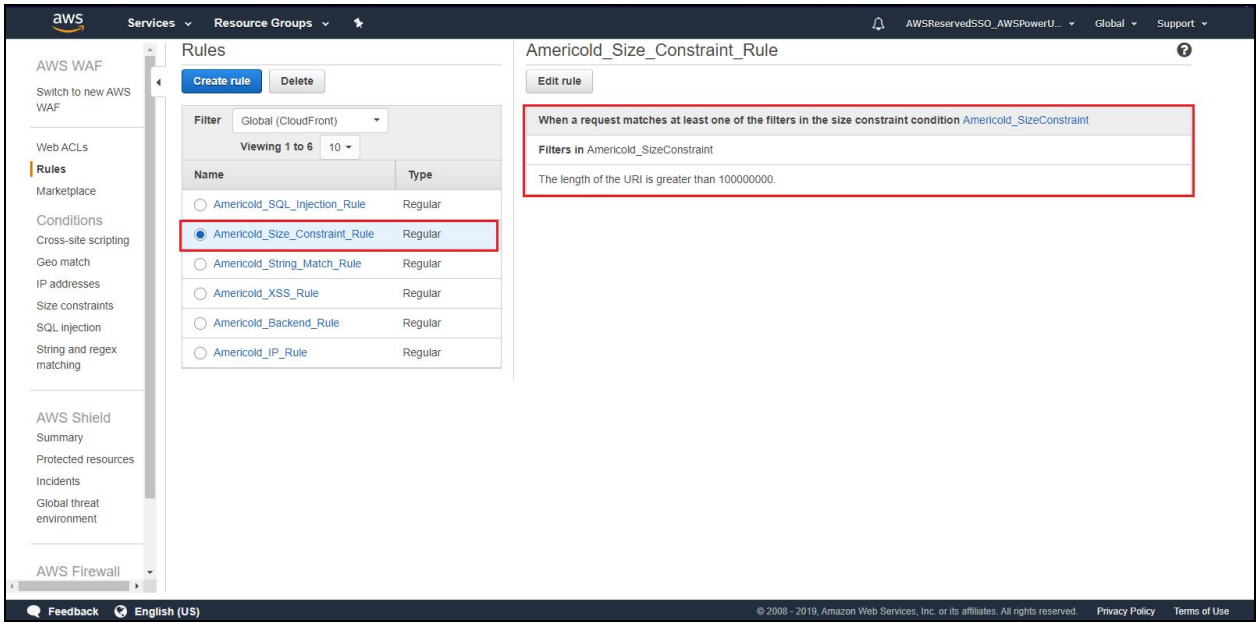
Now configure the name as **Americlod_IP_Rule,** CloudWatch metric name will be the same as the name of the rule, Select Role Type as **Regular Role** and region as default. Now coming to **Add conditions** section and add the **conditions** as follows.



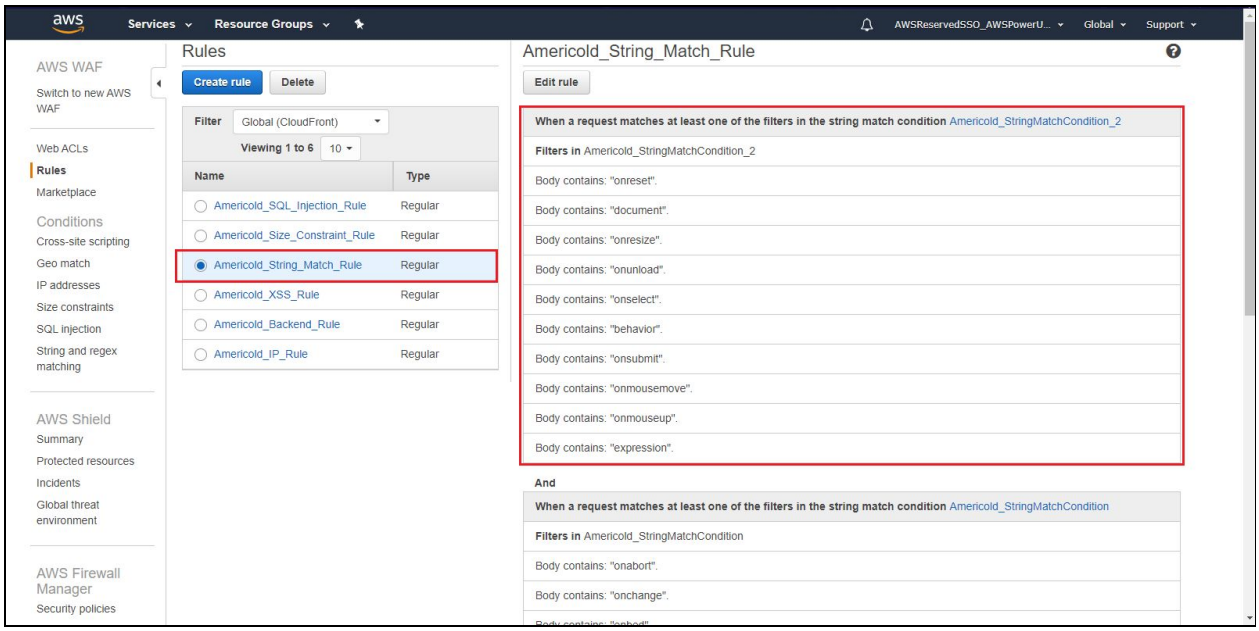
Following is the rules for **Americold_SQL_Injection_Rule**.

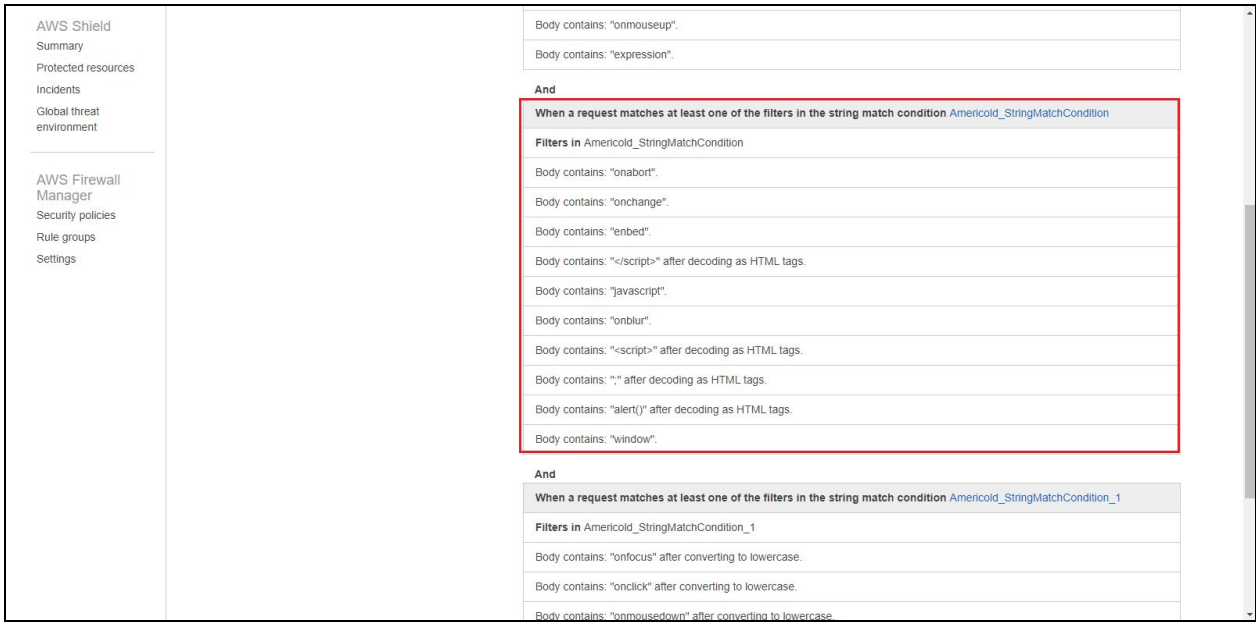


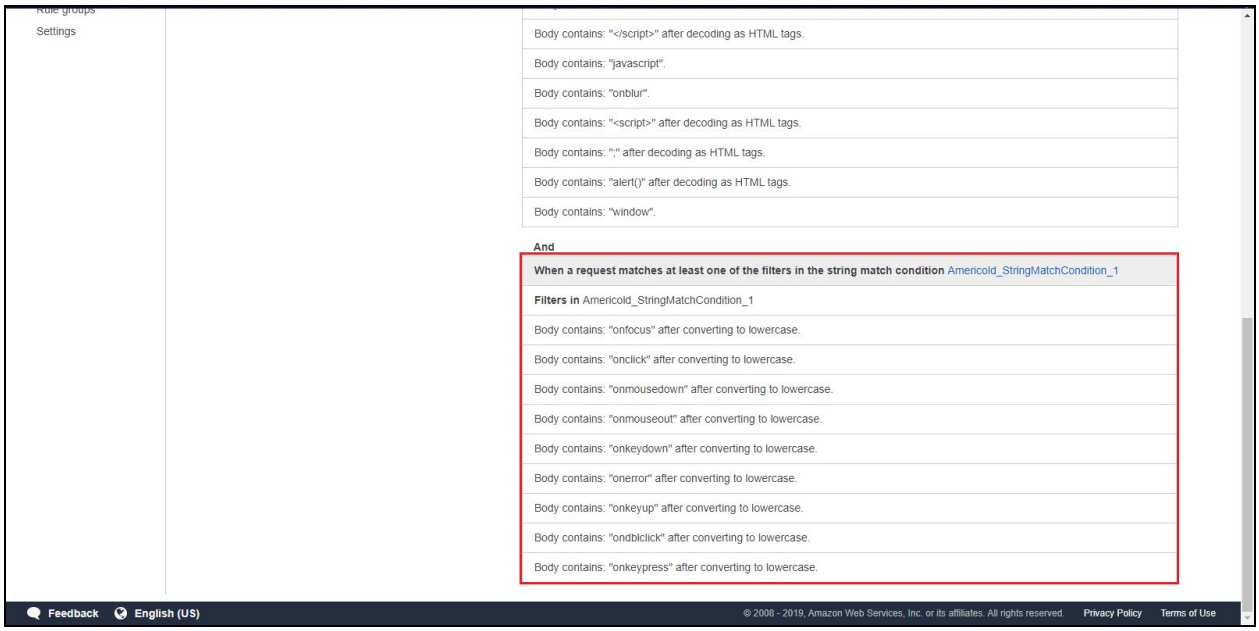
Following are the rules for **Americlod_Size_Constraint_Rule.**



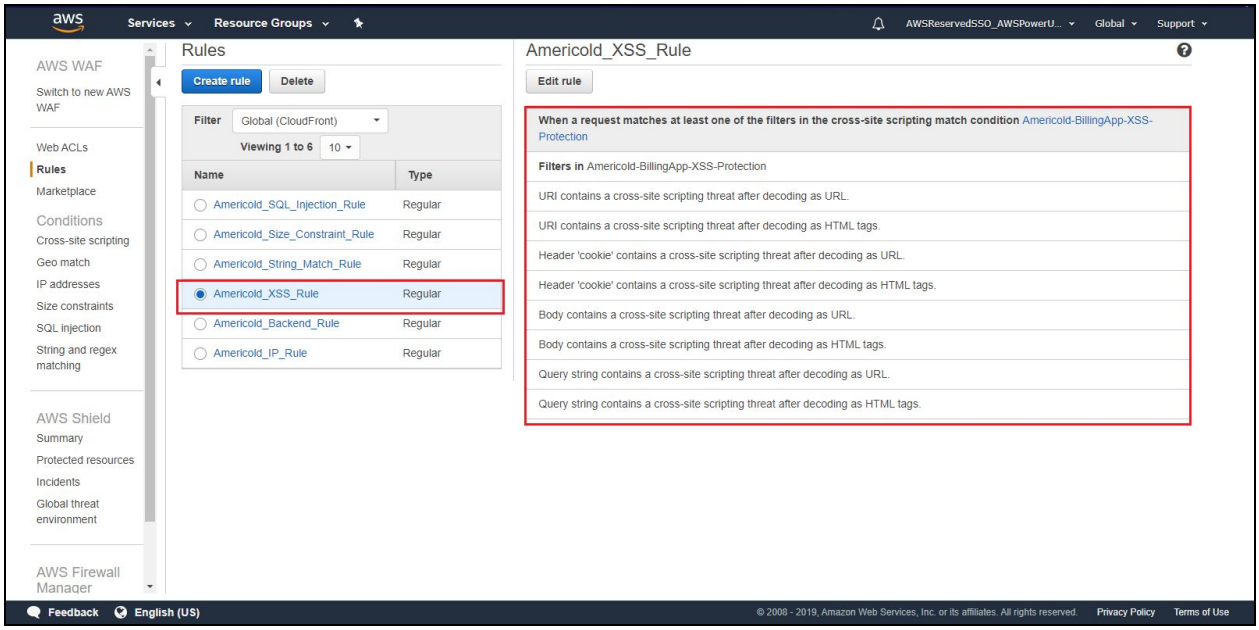
Following are the rules for **Americlod_String_Match_Rule**.



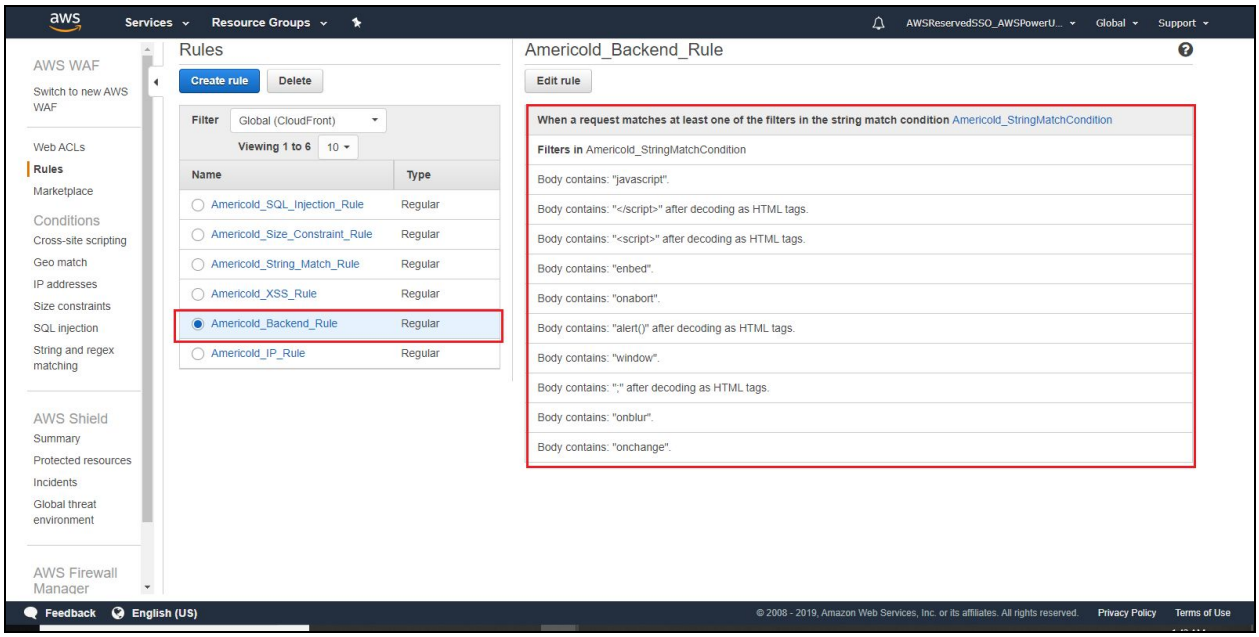

AWS Shield
Summary
Protected resources
Incidents
Global threat environment
AWS Firewall Manager
Security policies
Rule groups
Settings
Body contains: "onmouseup".
Body contains: "expression".
And
When a request matches at least one of the filters in the string match condition Americold_StringMatchCondition
Filters in Americold_StringMatchCondition
Body contains: "onabort".
Body contains: "onchange".
Body contains: "enbed".
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.


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

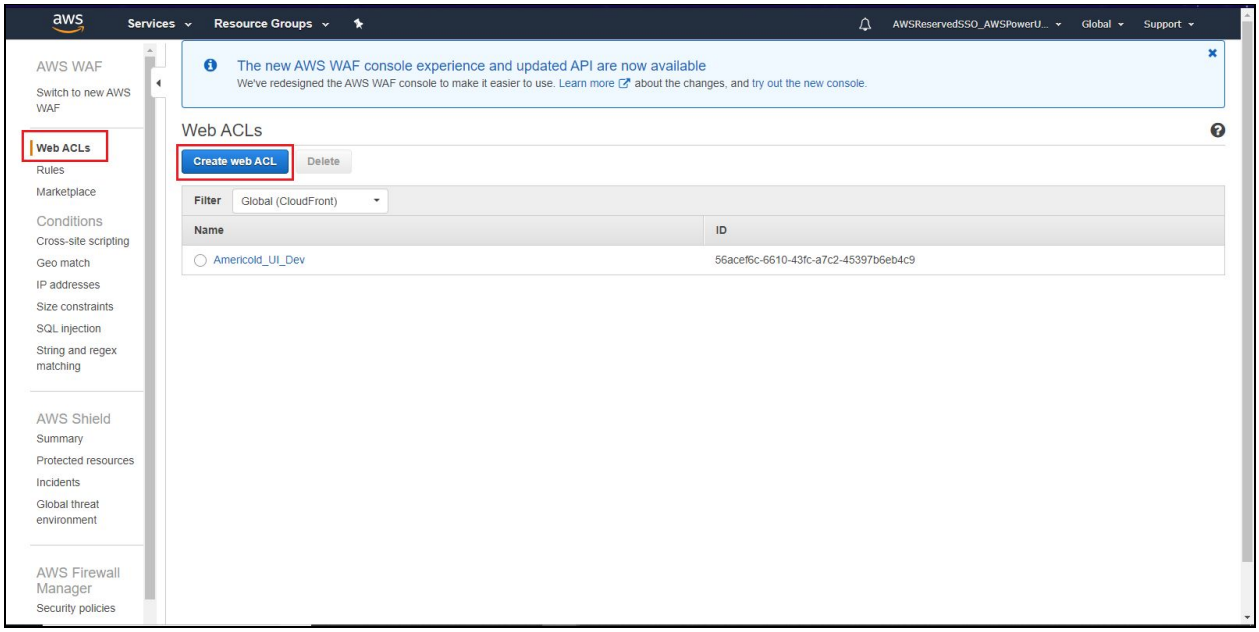
Following are the rules for **Americlod_XSS_Rule**.



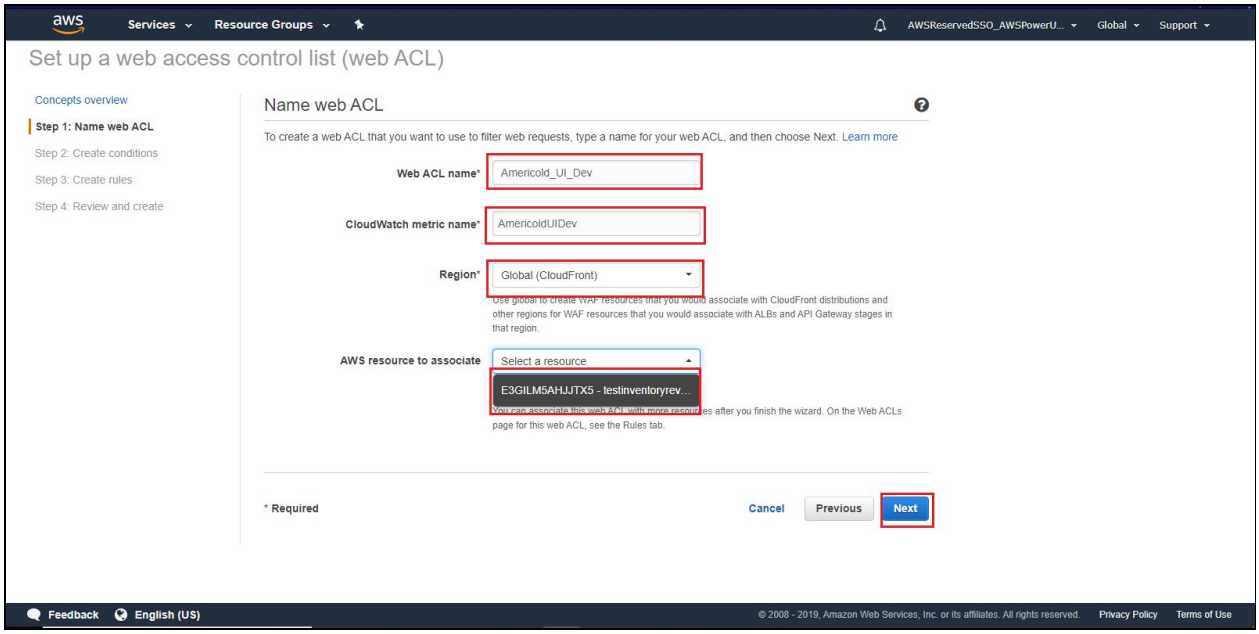
Following are the rules for **Americlod_Backend_Rule**.



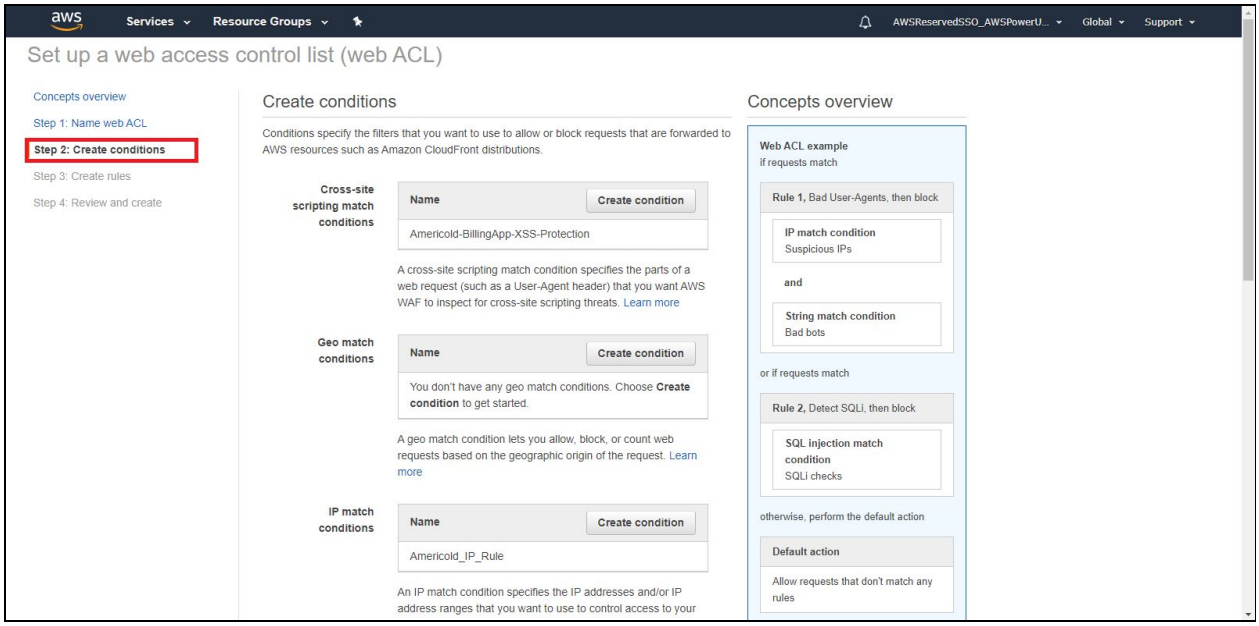
Now we have to go to **WebACLs** portal and click on **Configure web ACL**.



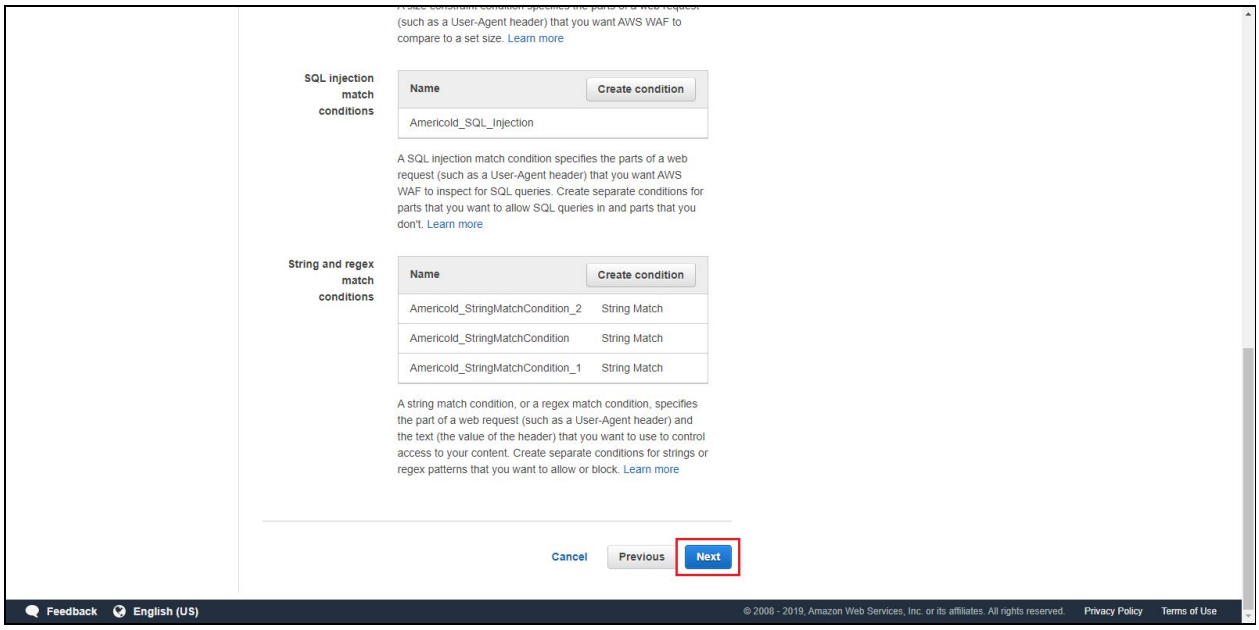
Now give a name to web ACL as **Americold_UI_Dev** and CloudWatch metric name will be automatically generated. Click on **Next**.



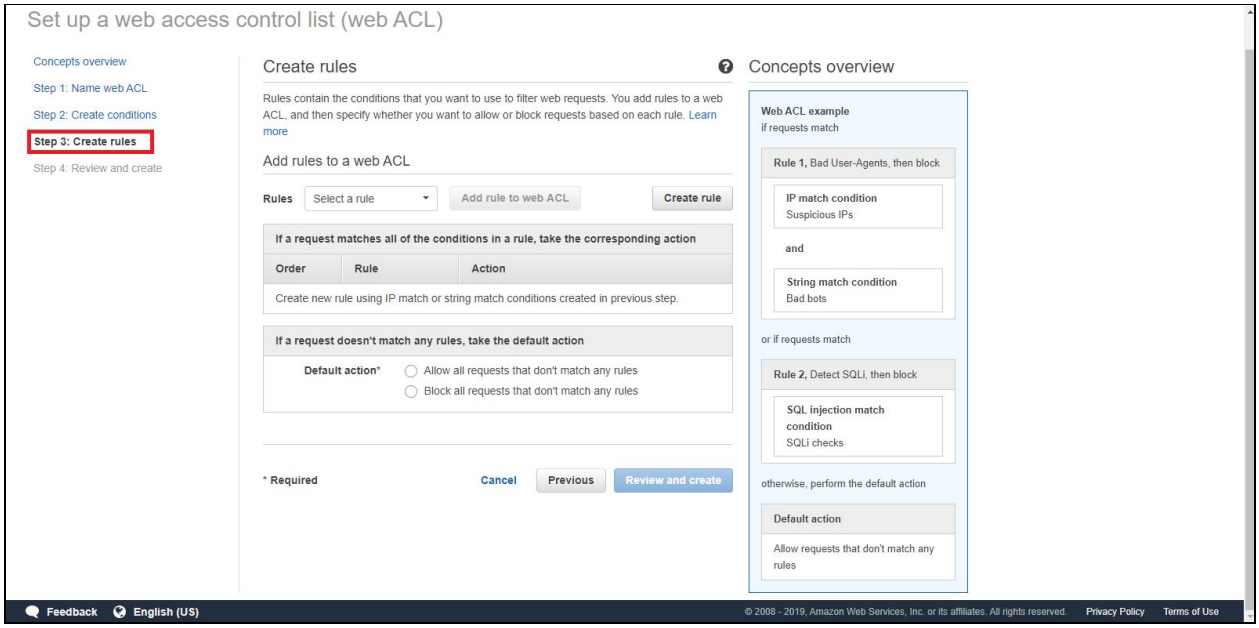
Now in the second step create conditions for Cross-site scripting match conditions and now click on **create condition**.



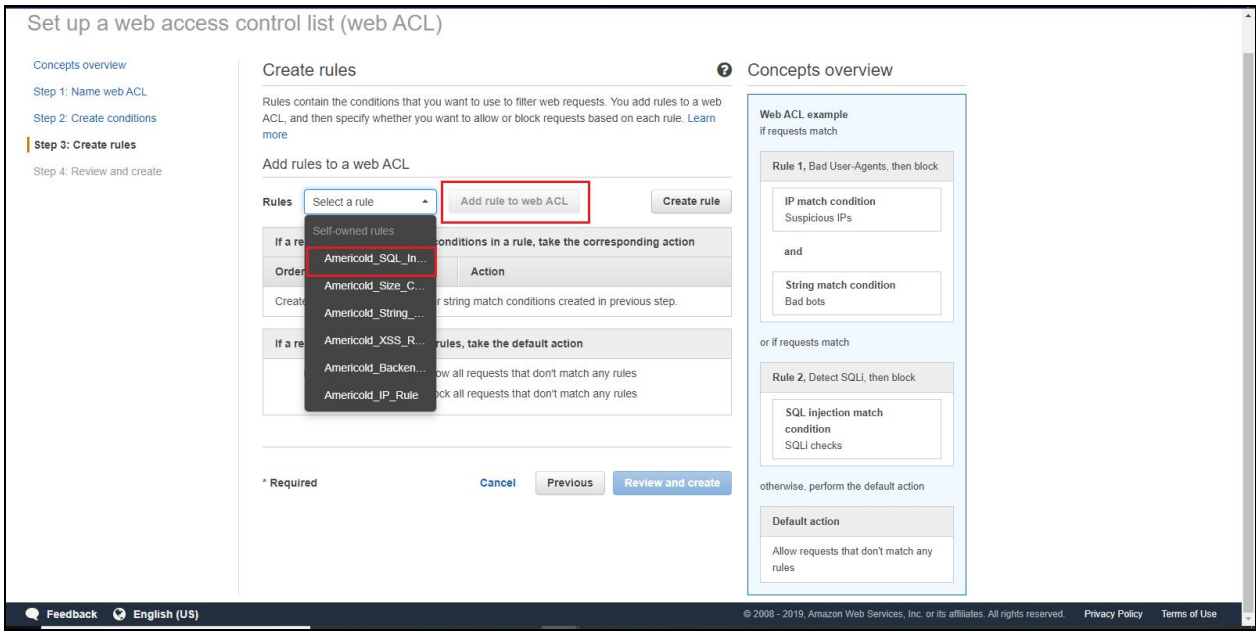
Click on **Next** button.



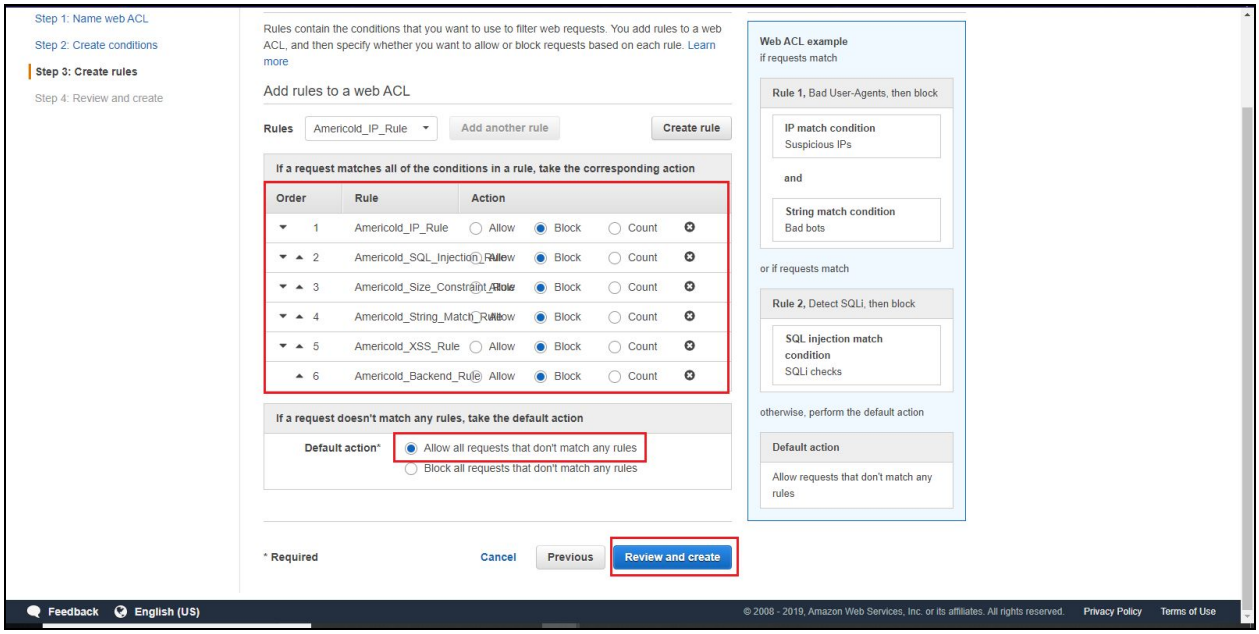
Now go to step 3 and create rules for web ACL.



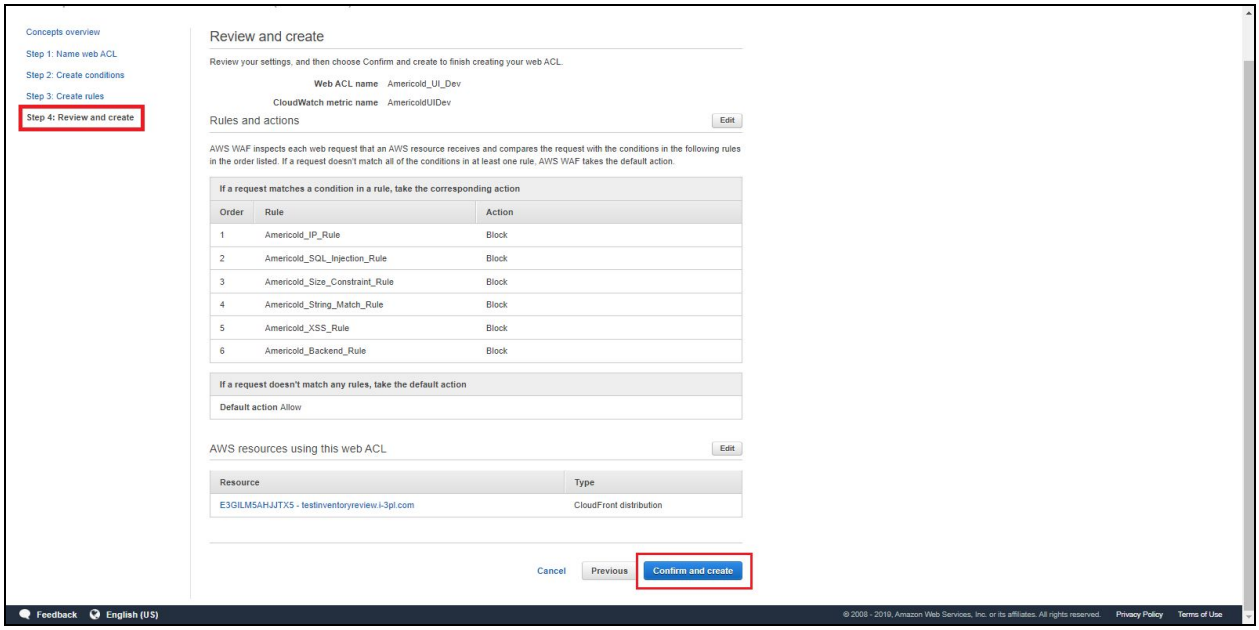
Select Rules that was created.



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

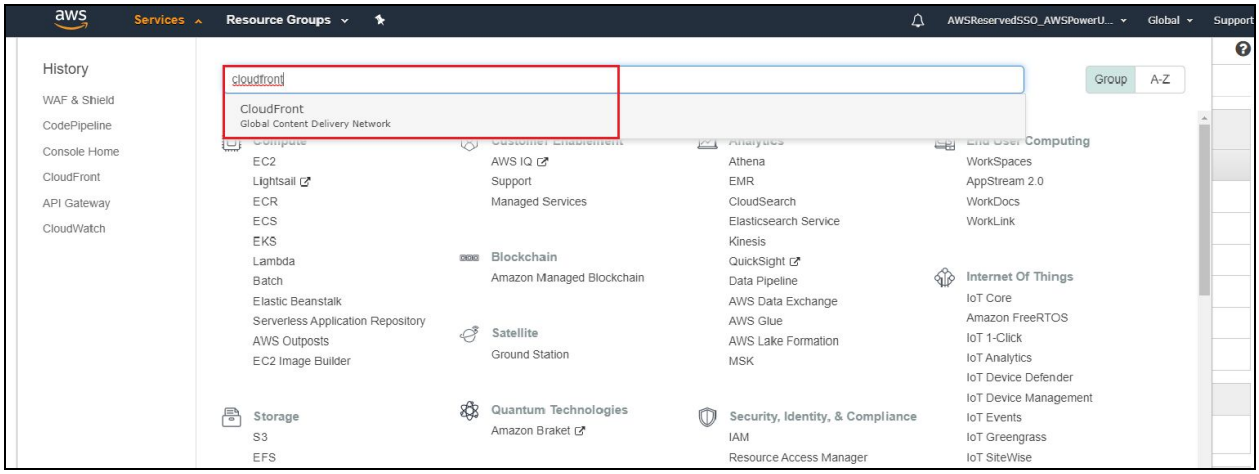


Now we will get into the step4, Click on Confirm and Create.

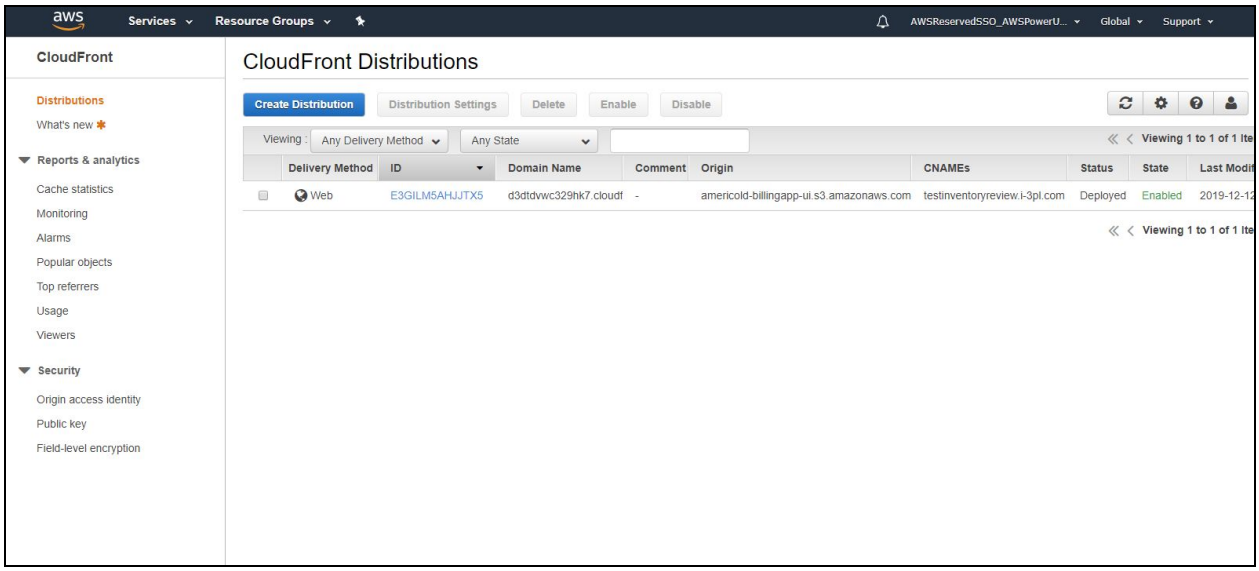


After adding rules to Web ACLs we need to got to AWS Cloudfront service to add our WAF rules to frontend application.

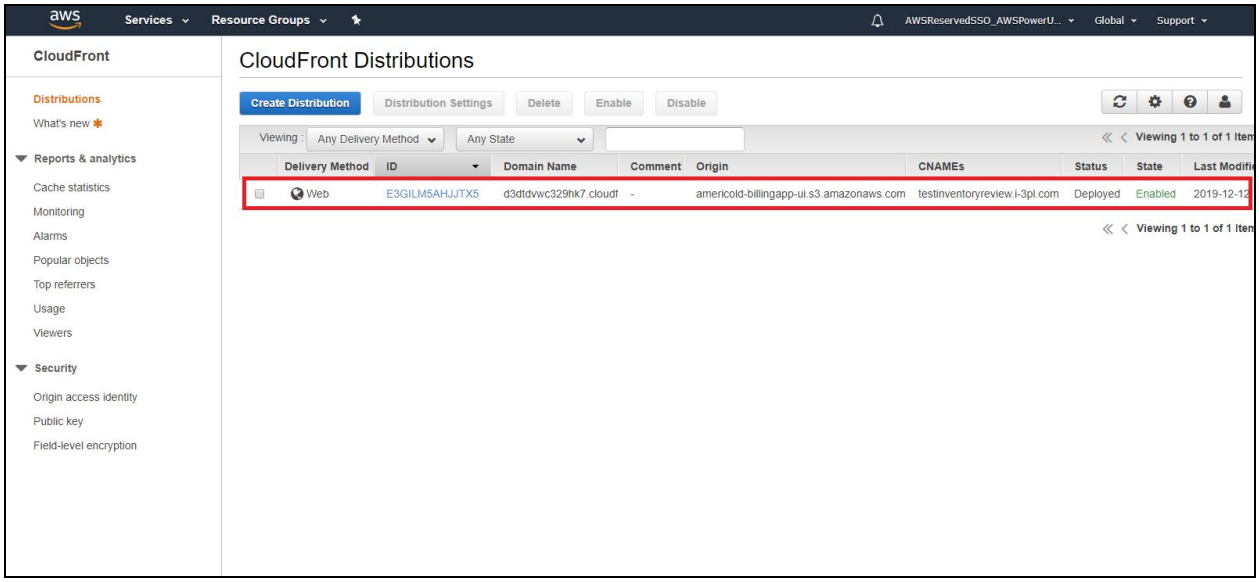
Now search for cloudfront service in console



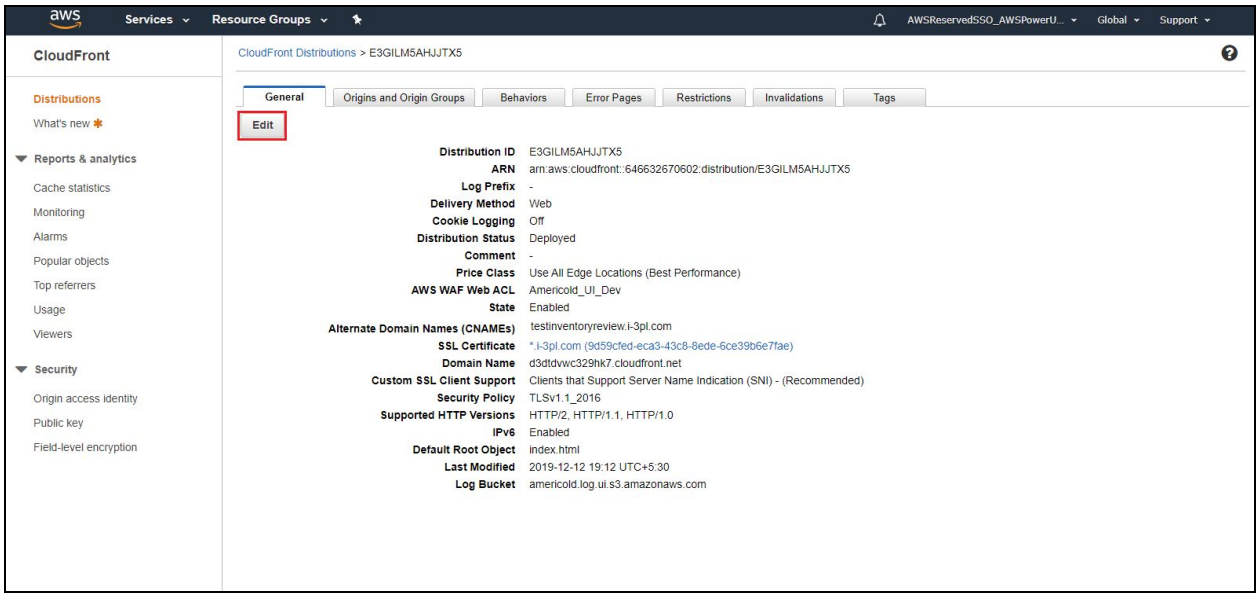
Now the dashboard will be as follows



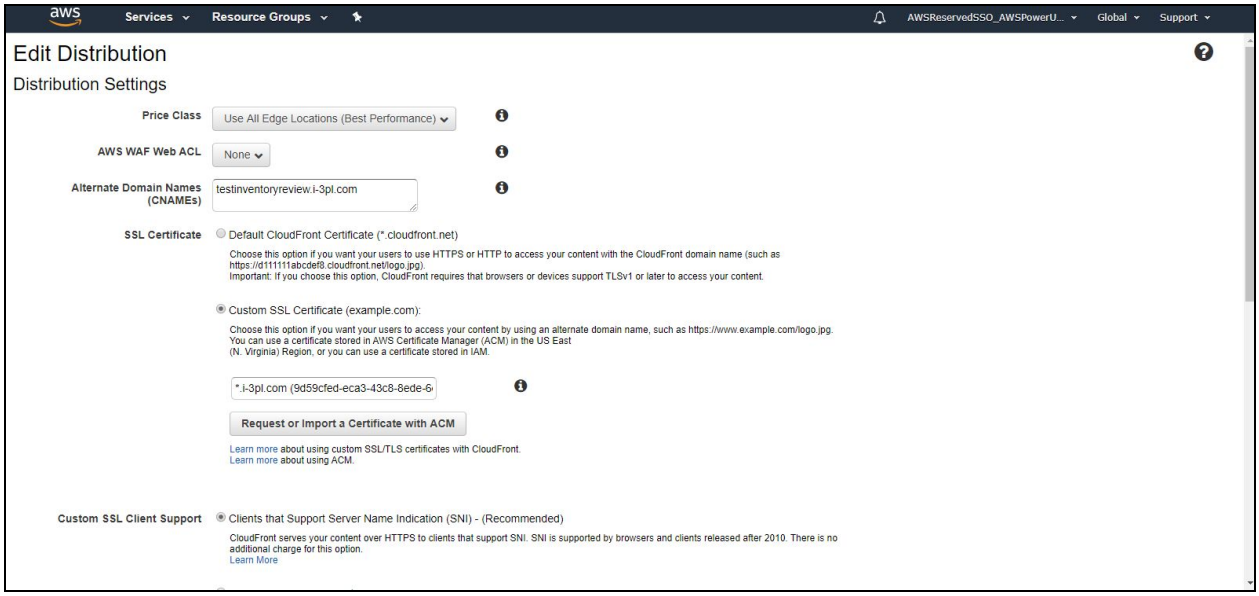
Now we are adding the WAF rules to already created Cloudfront distribution.



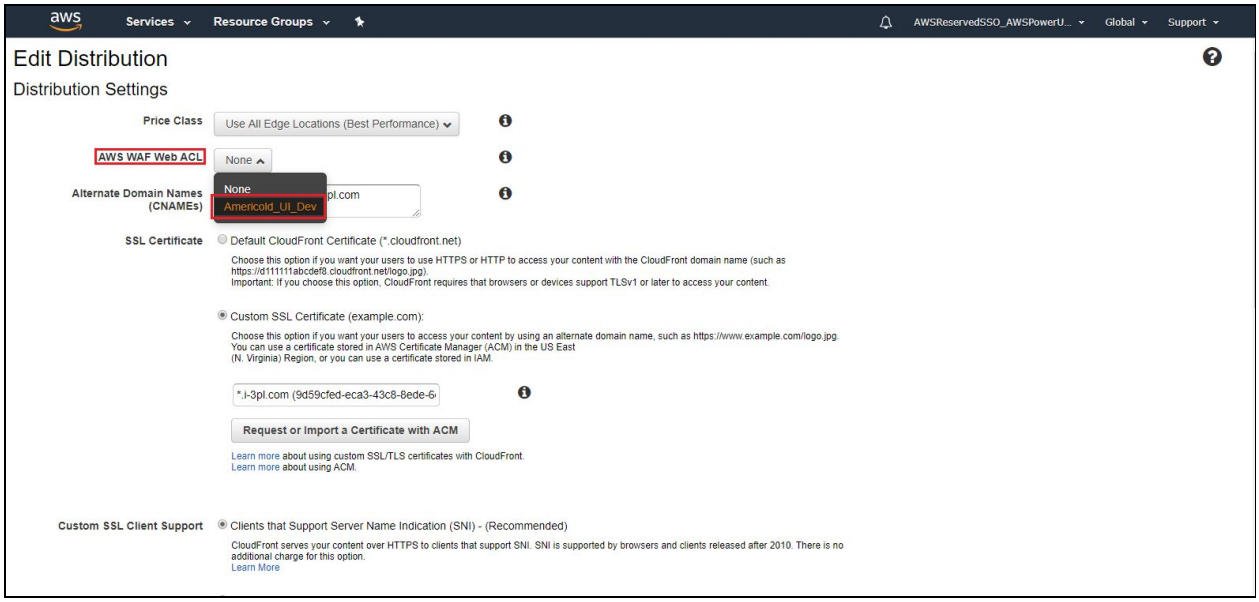
Now select the Cloudfront distribution and click on **edit**



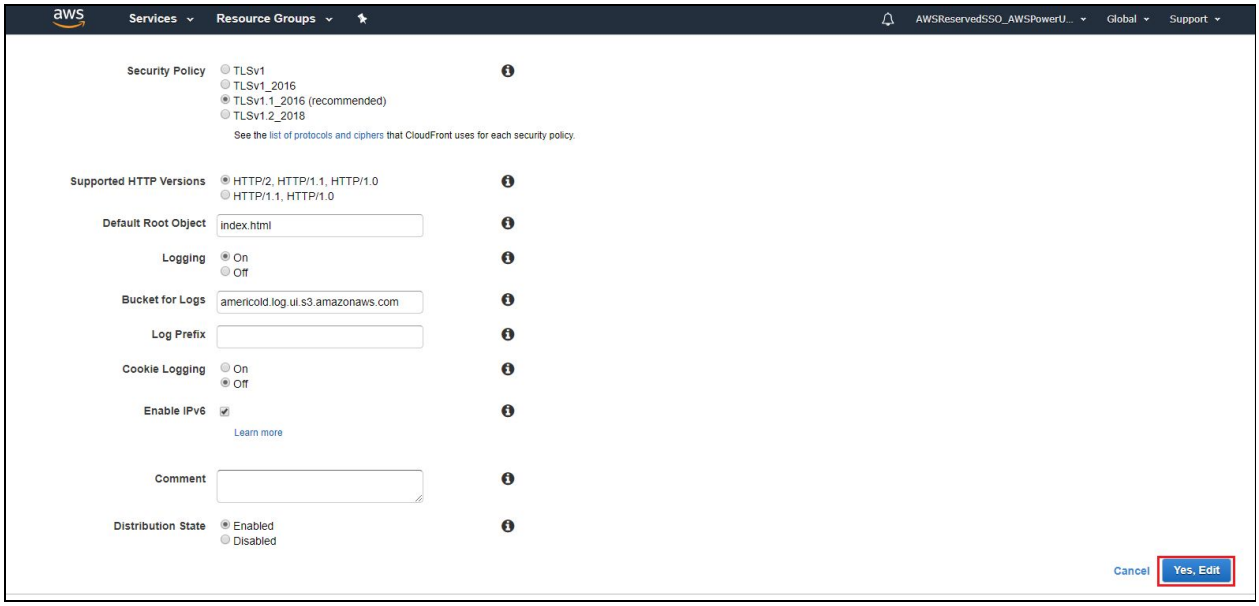
After click on the edit the dashboard is shown as below



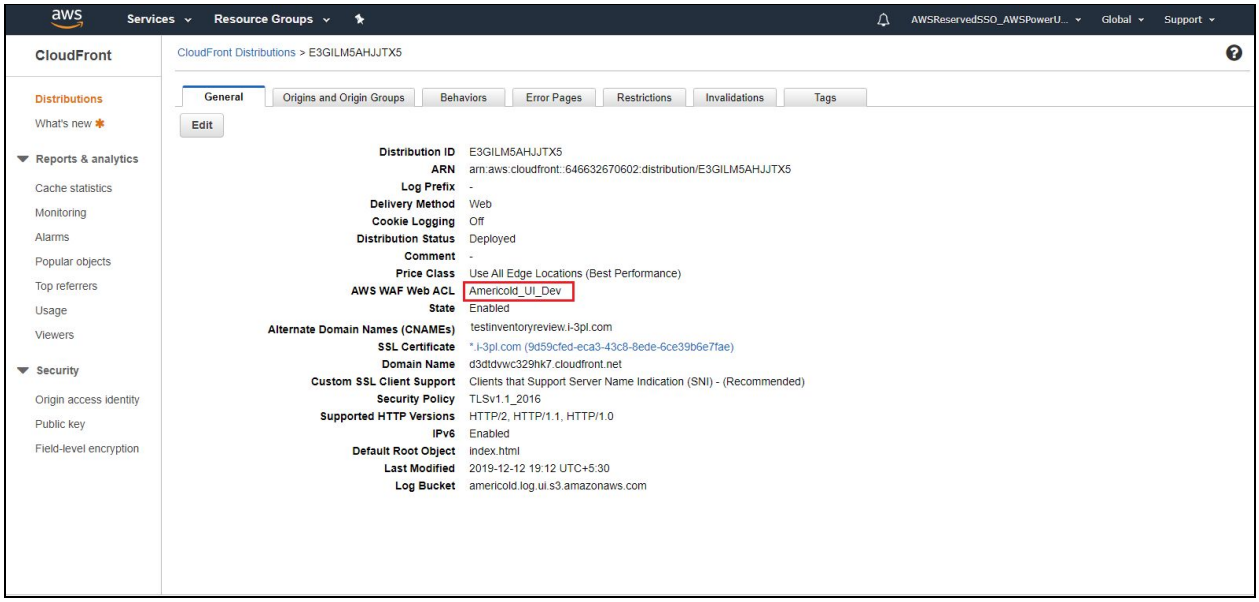
Now click on the drop down that was showing for **AWS WAF Web ACL** and select the Web ACL we have created in the above steps



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



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



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

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