Security in the cloud

Assignment-2

WAF (Web Application Firewall)

Assignment Brief

|  |  |
| --- | --- |
| Student Name/ID Number | Yazan Atary |
| **Course Title** | **Security in the cloud** |
| Academic Year | 2023/24 |
| **Assignment Title** | **WAF** |
| Issue Date | **4/18/2024** |
| Submission Date | **4/19/2024** |
| IV Name & Date |  |

|  |
| --- |
| **Submission Format** |
| The submission is in this form as clear screen shot under every question.  You can use this website to add your screen shots to the PDF file : <https://www.ilovepdf.com/blog/how-to-insert-an-image-into-pdf> |

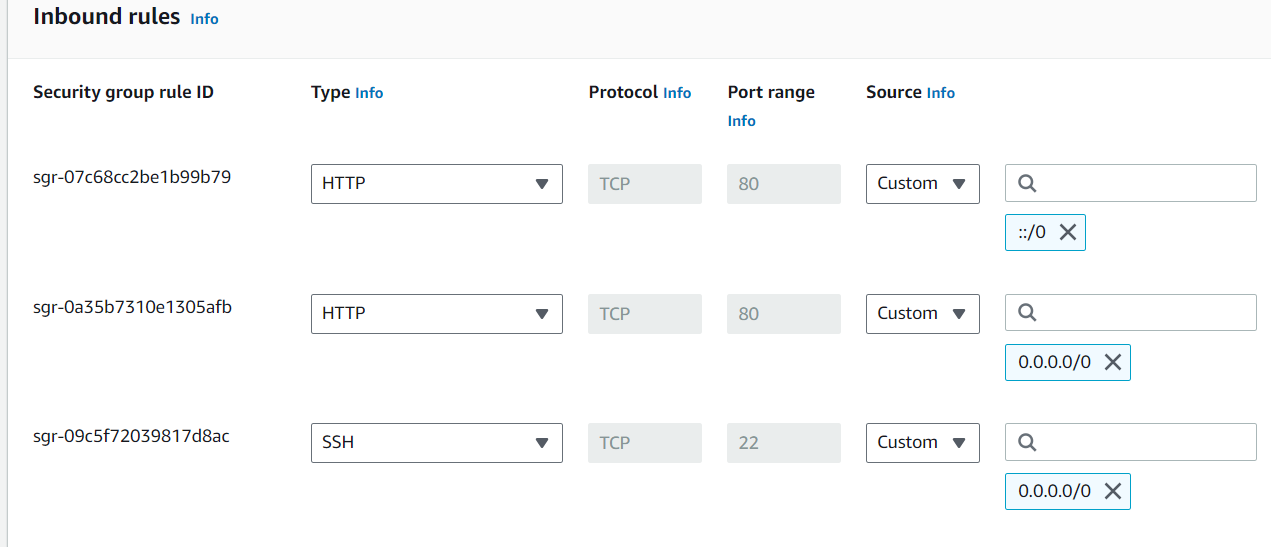
1

**Web Application Firewall (WAF)**

Web application firewall service that lets you monitor web requests that are forwarded to an Amazon API Gateway API, an Amazon Cloud Front distribution, or an Application Load Balancer. You can protect those resources based on conditions that you specify, such as the IP addresses that the requests originate from).

1. Create instance

* Name the instance.
* Go to EC2 service, launch instances.
* Choose AMI “amazon Linux”.
* Create new key pair (.ppk).
* Create security group.
* From network settings, add subnet as a1.
* Create instance.
* Modify your created security group as below.



* Create another instance, but add subnet b1, select existing security group.
* Open putty, and add the IP address for your 1st instance and use the same key pair.
* After your CLI is opened, start copying the below commands:

**#/bin/bash**

**sudo su**

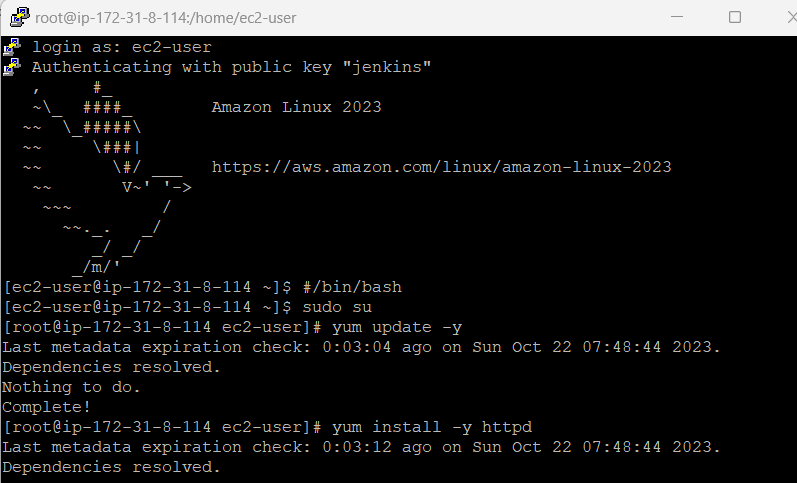
**yum update -y**

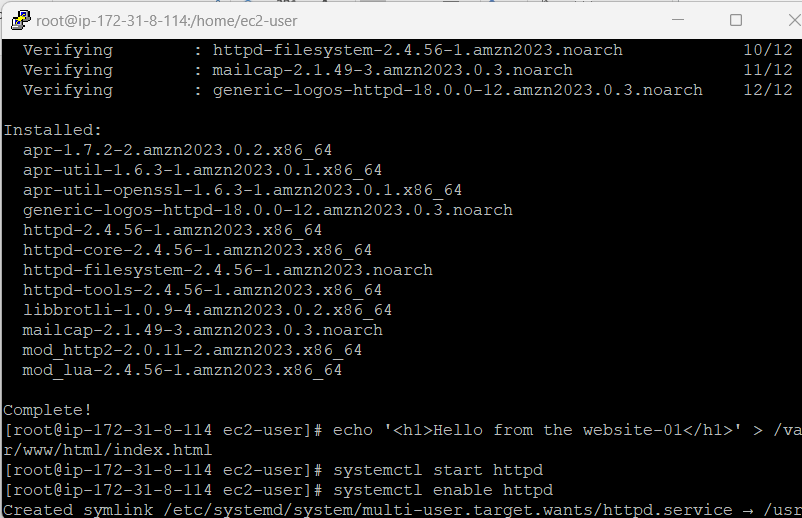
**yum install -y httpd**

**echo '<h1>Hello from the website-01</h1>' > /var/www/html/index.html**

**systemctl start httpd**

**systemctl enable httpd**





3

* Do the same for your 2nd instance , copy the IP address for 2nd instance and open putty , start copying the below commands :

**#/bin/bash**

**sudo su**

**yum update -y**

**yum install -y httpd**

**echo '<h1>Hello from the website-02</h1>' > /var/www/html/index.html**

**systemctl start httpd**

**systemctl enable httpd**

* Initiate a request from your web browser to <http://YOUR_MACHINE_IP_ADDRESS>

Provide a screenshot

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

* Do the same for the 2nd instance.

Provide a screenshot

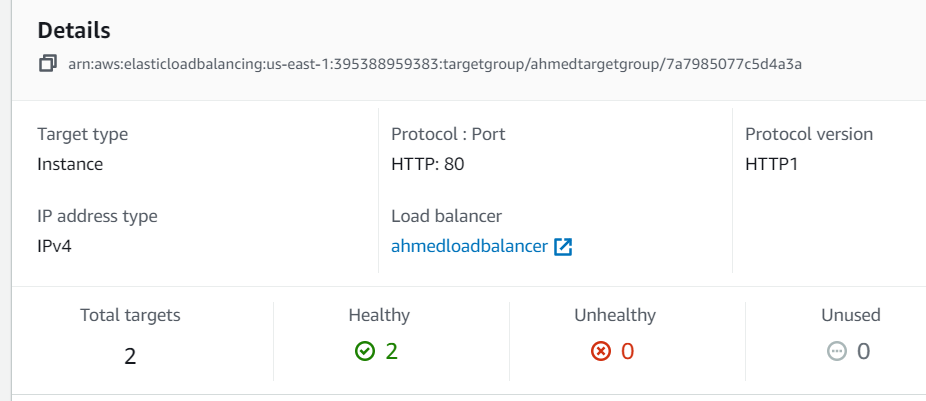
A screen shot of a computer program

Description automatically generated

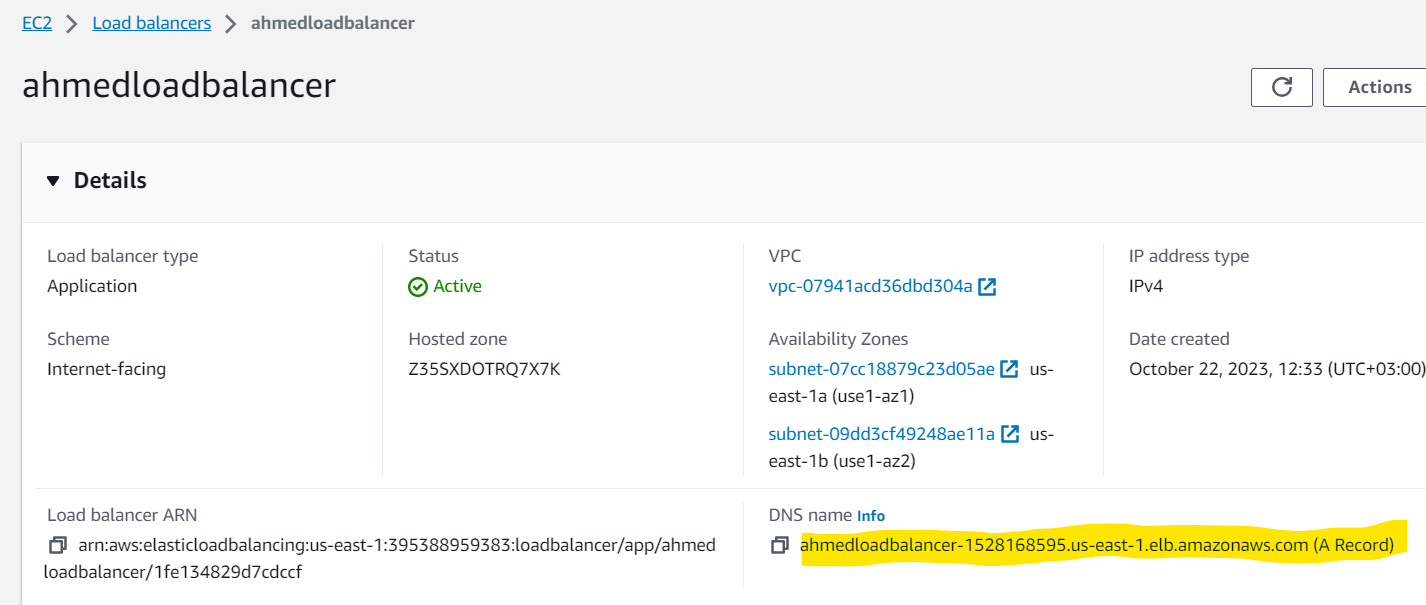


1. Create a Load Balancer

* Go to load balancer.
* Press on create load balancer, choose “**application load balancer**”.
* Name your load balancer.
* Select 2 used AZ’s, use existing security group.
* Create target group, choose target group type“instance”.
* Name the target group, then next.
* Select 2 registered targets for 2 used AZ’s.
* Include as pending below.
* Create target group.
* Open created target group, scroll down, select both targets, and register targets.



* Go to load balancer page, then select created target group.
* Create load balancer.
* Open the created load balancer and wait till the status be active then copy DNS name.



* Initiate a request from your web browser to <http://DNSName>. You should receive, as shown, by still refreshing the same page and notice that you should get two responses for your websites that are working on your DNS Name on ALB.

Provide two screenshots, showing the two web pages after refreshing the page.

* Explain the importance of application load balancers, illustrating how they contribute to the **resiliency** of your web application.

A screenshot of a computer

Description automatically generated

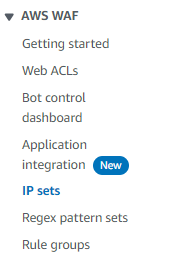
A close up of a computer screen

Description automatically generated

**Application Load Balancers (ALBs)** act as the adept traffic controllers of a web application. They efficiently distribute incoming traffic among various servers to prevent any one server from becoming overloaded. ALBs maintain the robustness and resilience of the web application by seamlessly managing numerous user requests. Like vigilant guardians, they quickly divert traffic away from underperforming servers to ensure continuous availability. Essentially, ALBs play a crucial role in ensuring that the web application not only functions smoothly but also stands resilient against digital challenges, thereby providing a consistent and reliable online experience.

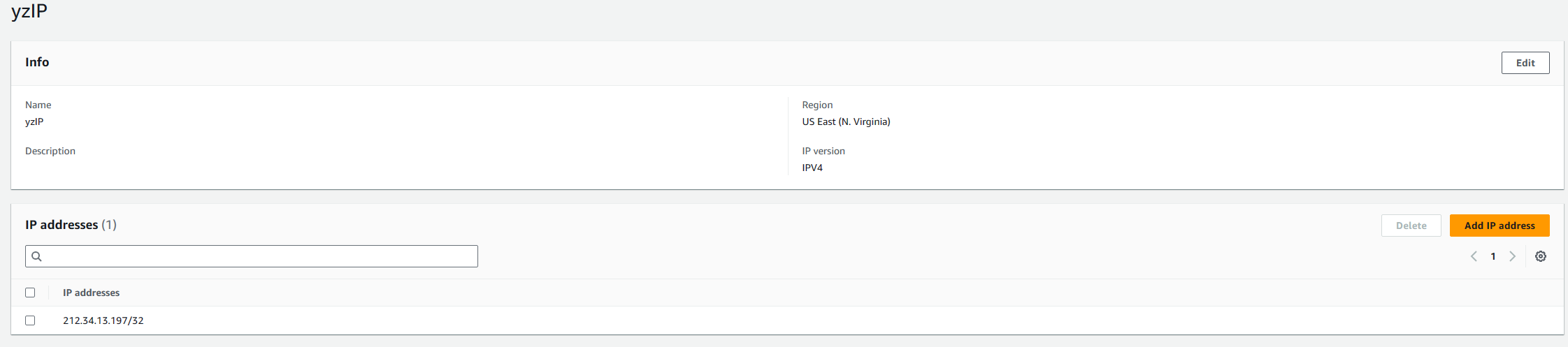
1. Create WAF on AWS

* Go to WAF service
* Go to “**IP sets**” from the left side, and press on “**Create IP set**”

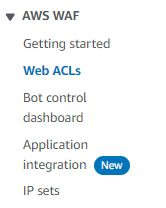


* Name your set, and add your public IP address in CIDR format (with your subnet mask).

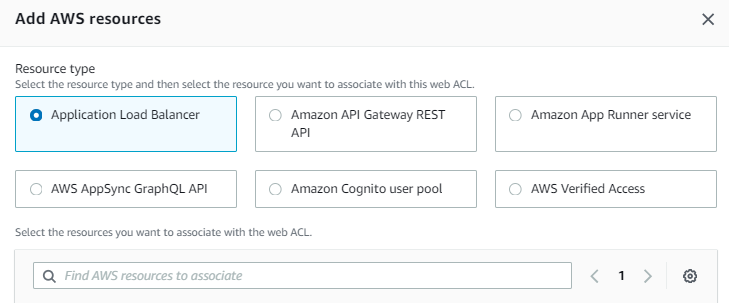
Provide a screenshot of your created IP set



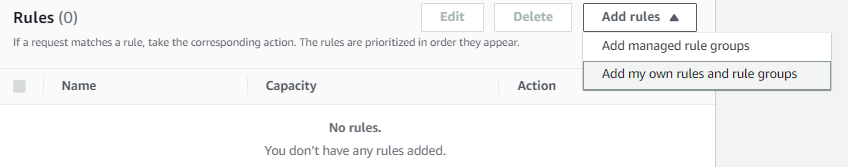
* Now, we are going to create a web ACL.
* Go to Web ACLs, and press on “**Create web ACL**”



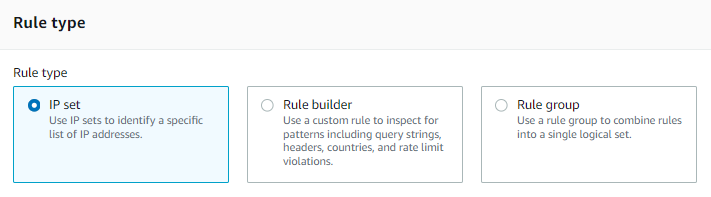
* Name your ACL.
* From “add AWS resources”, choose Application Load Balancer”.



* Search and add your previously created load balancer.
* Next, we have to add some rules to our web ACL. Choose “Add my own rules”.



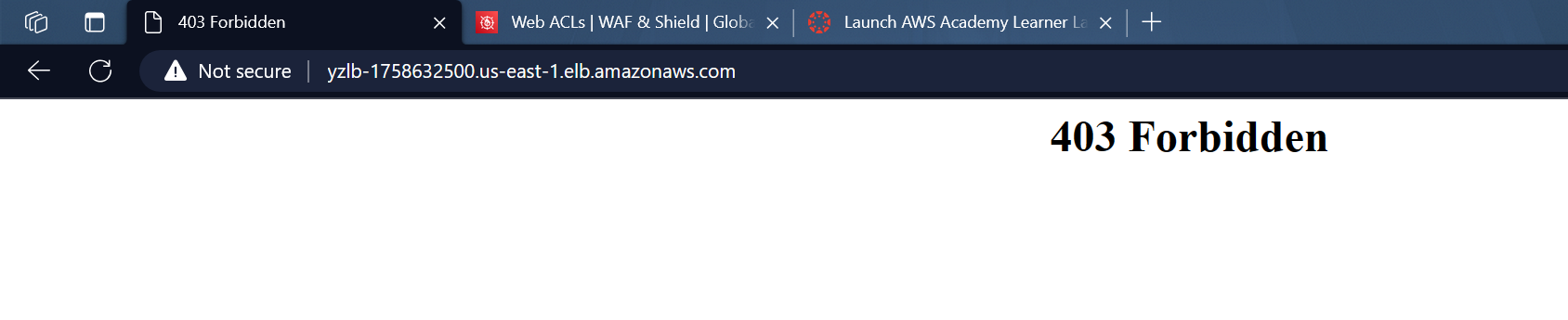
* Select the “IP set” rule type.



* Name the rule, and put the IP set that you have created before.
* The **first action** that we are going to do is to block the traffic from the addresses listed in the specified IP set. So, choose the **block** action and add the rule.
* In the configuration metrics, keep all other configurations at the default, and create your web ACL.
* Now, initiate a request from your web browser to the domain name of your <http://DNSName> and see whether you can access the URL, or not.

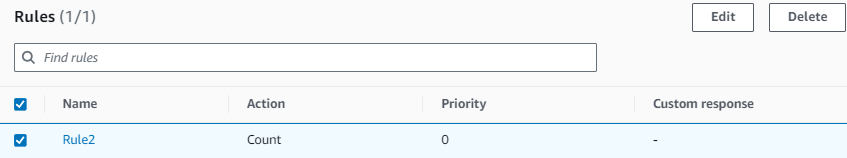
Provide a screenshot.

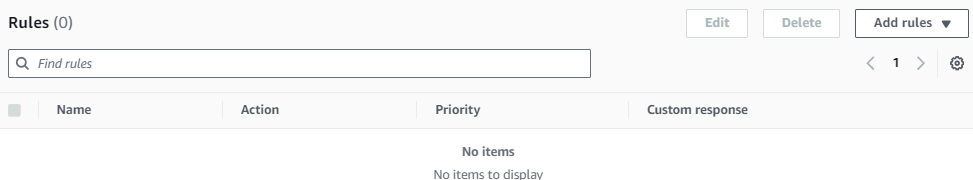
* Explain in detail the results you have obtained and why we would need to block requests from a specific range of IP addresses.

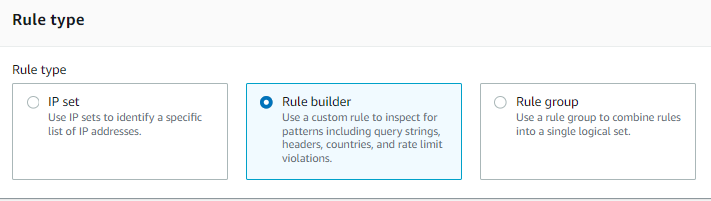


**When I set up a rule to block requests from my IP address, the result was immediate: a 403 forbidden error, signaling access denied. This action was a key security measure. Blocking requests from specific IP addresses, including my own, acts like a bouncer at a club's entrance, ensuring only authorized individuals can enter. In the realm of cybersecurity, this strategy is crucial for defending against potential threats. By limiting access from certain IPs, we create a security barrier, protecting our web application from malicious actions and potential cyber threats. Essentially, it's like putting up a "No Entry" sign for those we don't want inside, safeguarding the integrity and confidentiality of our digital environment.**

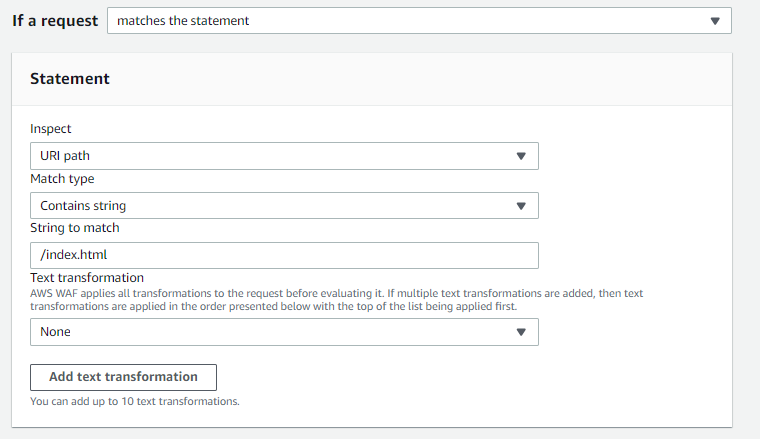
* Now, we are going to delete this rule and try to use another actions. Go to the web ACL that you have created and access the rules that you have configured.
* Specify the rule that you want to delete and press on delete.



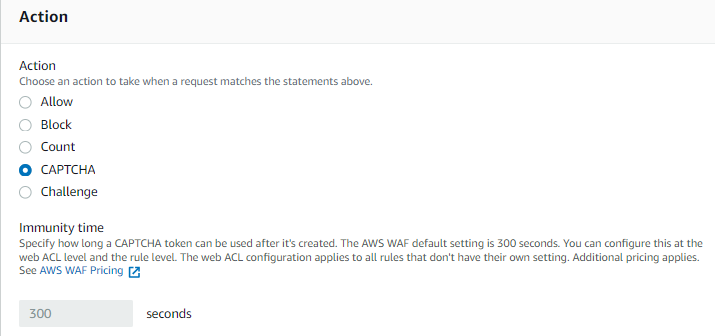
* The **second action** that we are going to try is the CAPTCHA test that is used on websites to determine whether the user is human or automated software.
* After deleting the previous rule, we are going to create a new rule. 
* Press on “Add rules”, then choose “Add my own rules” option.
* Select the “Rule builder” rule type.



* Name the rule, and choose “Regular rule”.
* Add the following details to the matching criteria.

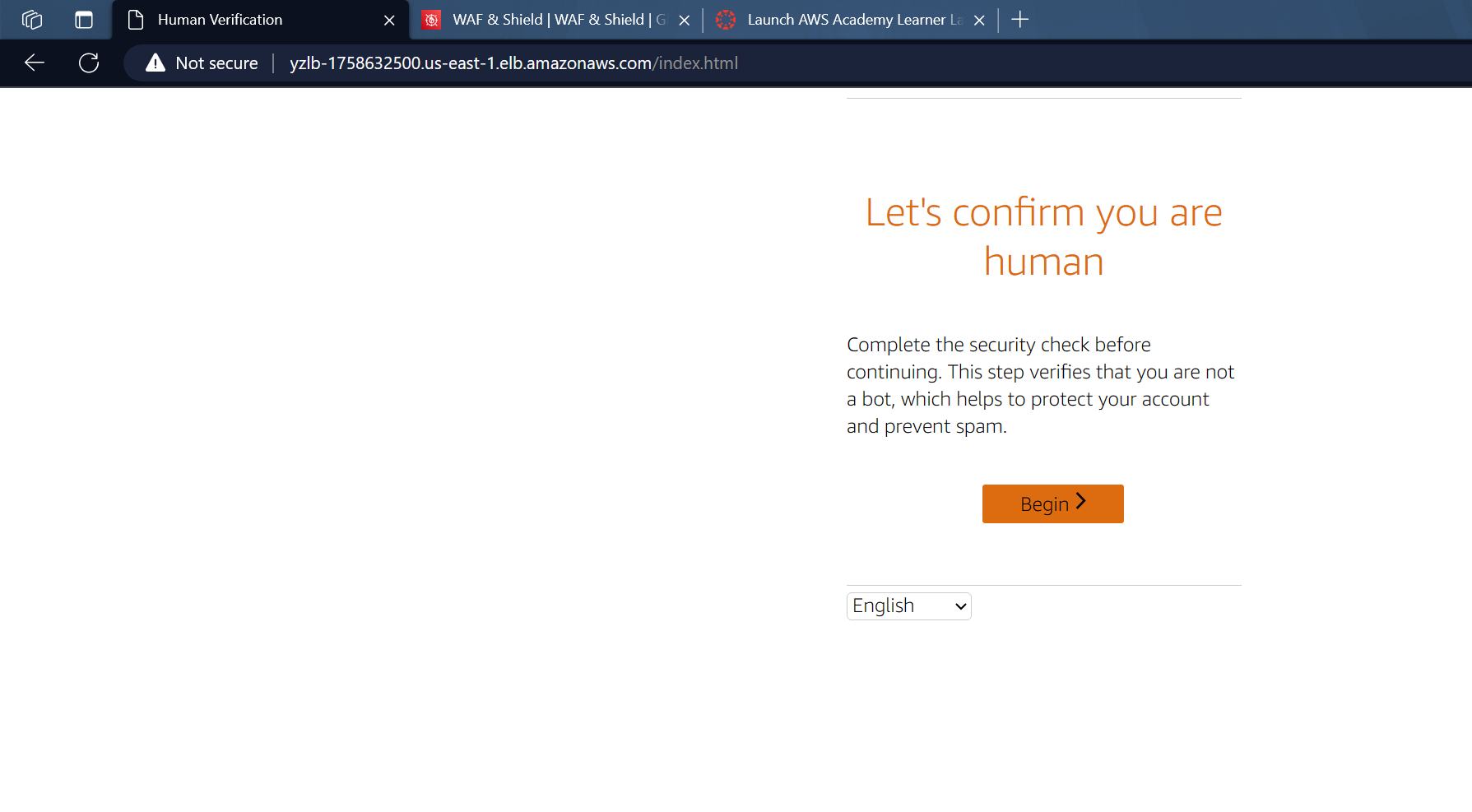


* Now, the action that we want to do is the CAPTCHA.



* Add the rule when you finish setting your options.
* Now, let’s test our rule!
* Initiate a request from your web browser to the domain name of your <http://DNSName/index.html> and observe the results.

Provide a screenshot.



* Explain in detail the results you have obtained and why we would need to use CAPTCHA within our rule,
* **Why Use CAPTCHA in Rules:**
* Incorporating CAPTCHA into security rules serves two critical purposes, enhancing the overall security posture of a web application.
* **1-Better Security:** CAPTCHA functions as an additional layer of security by ensuring that incoming requests originate from genuine human users rather than automated bots. It leverages challenges that are easy for humans to solve but challenging for automated scripts, thwarting illegitimate access attempts.
* **2-Guard Against Attacks:** CAPTCHA is a valuable defense mechanism against various automated attacks, such as spam and efforts to compromise the system using scripted tools. By presenting a human-verification challenge, it effectively filters out malicious automated activities, bolstering the application's resilience.
* as well as the security impact on our web application services.
* **Security Impact on Web Application Services:**
* The inclusion of CAPTCHA within security rules contributes significantly to the overall robustness of web application services.
* **1-Boosted Defense:** By integrating CAPTCHA, the defense mechanisms of the web application are fortified. CAPTCHA serves as a proactive measure, requiring users to prove their human identity before accessing certain features, adding an extra layer that potential attackers must navigate.
* **2-Less Vulnerable:** The implementation of CAPTCHA renders the web application less susceptible to breaches. It introduces complexity for automated systems attempting unauthorized access, reducing the vulnerability of the application to malicious exploitation.
* Additionally, mention an attack scenario that could be prevented when using CAPTCHA.

**Attack Scenario Prevented:**

CAPTCHA plays a pivotal role in thwarting a specific type of cyber threat: brute force attacks.

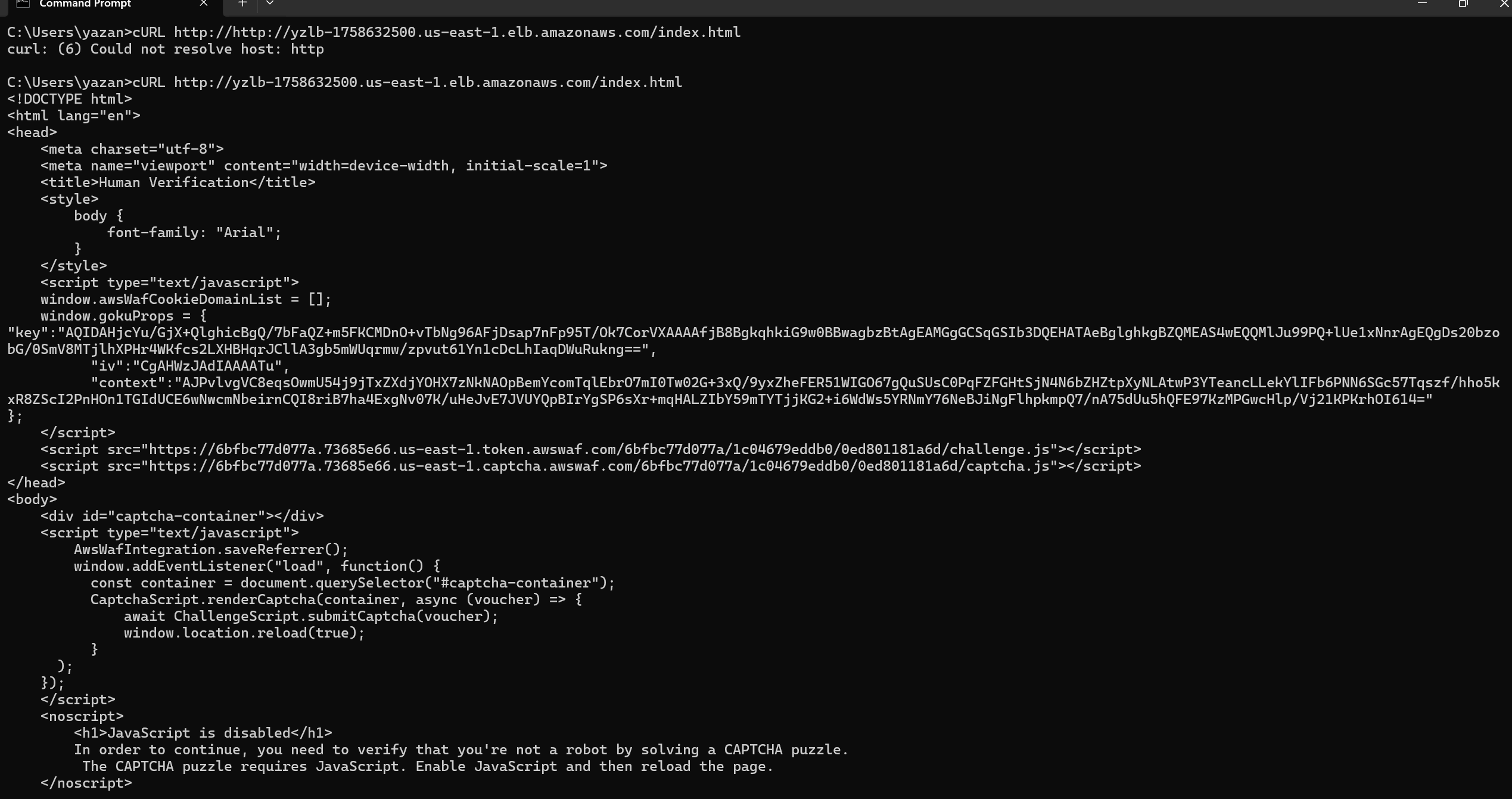
**Stopping Brute Force Attacks:** In scenarios where attackers employ brute force tactics, attempting numerous combinations to gain unauthorized access, CAPTCHA acts as a formidable obstacle. The human verification step introduced by CAPTCHA makes it considerably more challenging for automated tools to persistently probe the system, thereby significantly improving the overall security of the web application.

* Now, let’s try to use the cURL tool to initiate a request to our web server
* Use the following command on your windows CLI and observe the results

**curl "**[**http://DNSName/index.html**](http://DNSName/index.html)**"**

Provide a screenshot

* Explain the results that you have observed (try to see the results before and after setting the rule)



**I modified the rule in WAF ACLs to force captcha for requests from my IP.**

**This means that when you submit the request, you face a captcha challenge before accessing the content to ensure that.**

**Similar to the blocked step, at this point, the cURL command only fetches the HTML code until the captcha challenge.**

We've completed our lab for today! I encourage you to delve deeper into the various options available on Web ACLs, such as query strings, challenges, and more.

Don't forget to clean up all used resources, including instances, target groups, load balancer, IP sets, and the web ACL.