ASAC – Security in the Cloud – Section IR1

Securing our Hosted Web Services

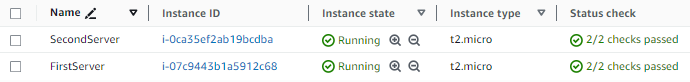
**Securing our Hosted Web Services**

During this lab session, we will host a website on a cloud server and design a secure solution to deploy this website in a resilient and secure manner. Additionally, we will establish the necessary security measures and resources to enhance the system's resilience. After that, we will create a test plan to conduct a comprehensive testing on the security countermeasure implemented in the cloud.

**Enabling resiliency solutions**

**Part 1: Hosting a website on a cloud server with load balancing**

* Create two EC2 instances and name them FirstServer and SecondServer
* As learned before (from the WAF lab), we are going to host our website on two different servers to take advantage of the load balancing feature. So, we will go through the same instance creation steps. You should have two created servers, as follows:



* Now, let’s upload our webpage on both websites. Connect with your EC2 instances and perform the following configuration commands (on each instance):

**#/bin/bash**

**#sudo su**

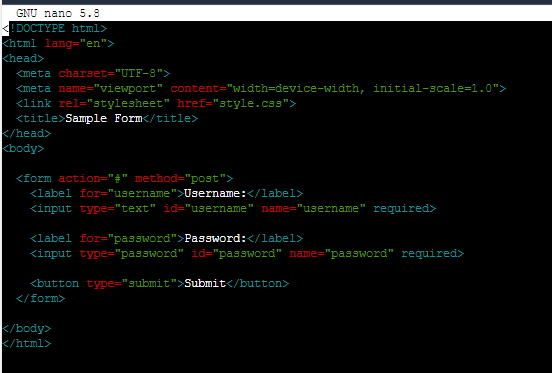
**#yum update -y**

**#yum install -y httpd**

**#cd /var/www/html/**

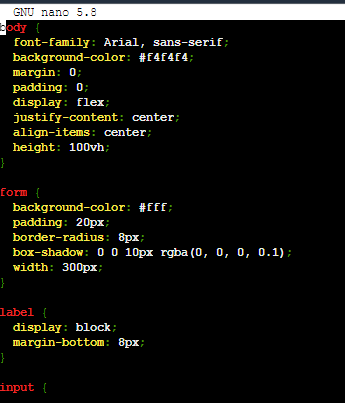
**#nano index.html**

**(Put the code of index.html and save the file, as follows)**



**#nano style.css**

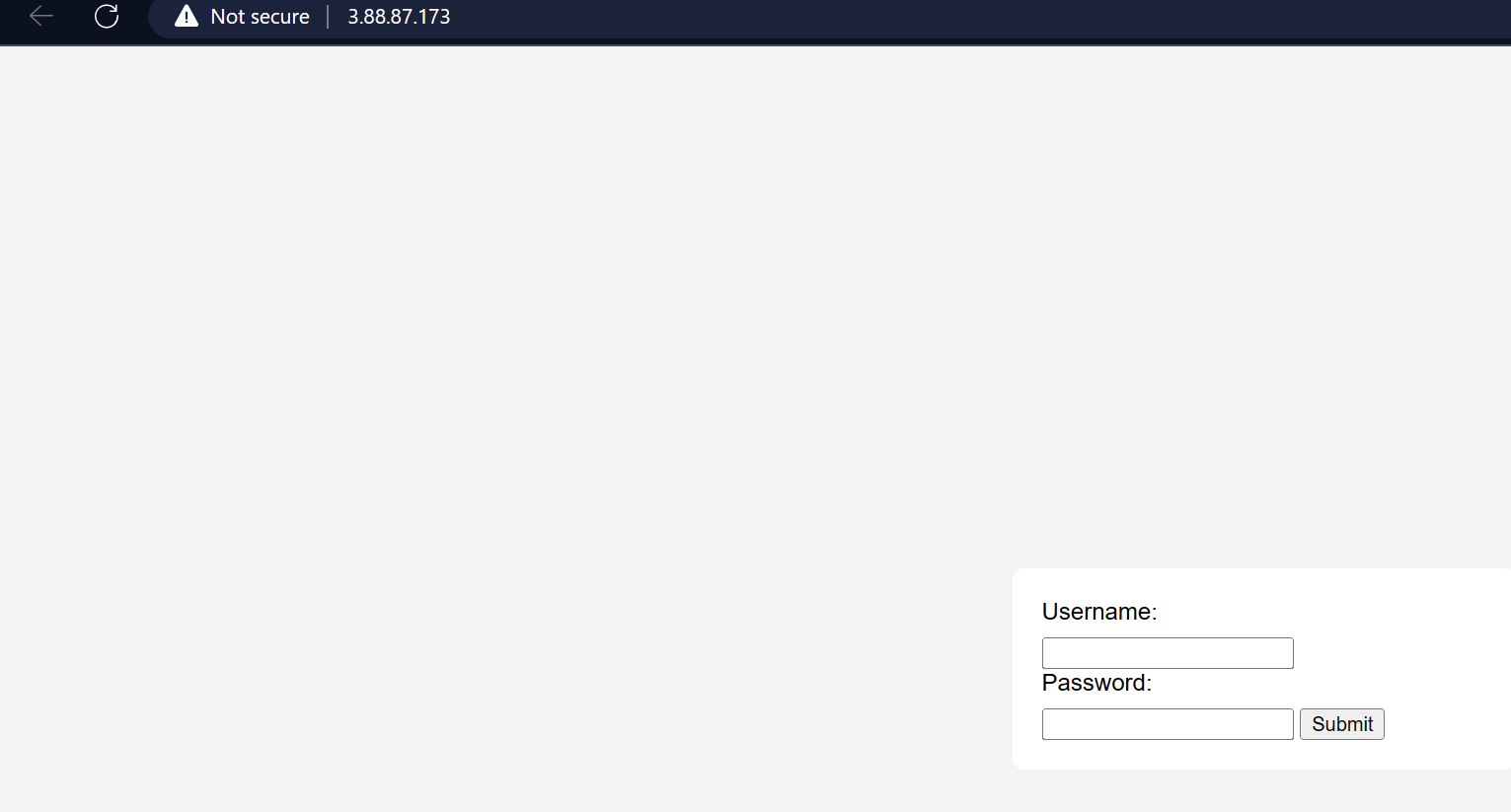
**(Put the code of style.css and save the file, as follows)**



**#systemctl start httpd**

* Do the same for your 2nd instance.
* Make sure that you access both websites and the web page appears clearly on the website

Provide a screenshot of the website that you opened on your web browser (Show the IP address of the machine form the URL)

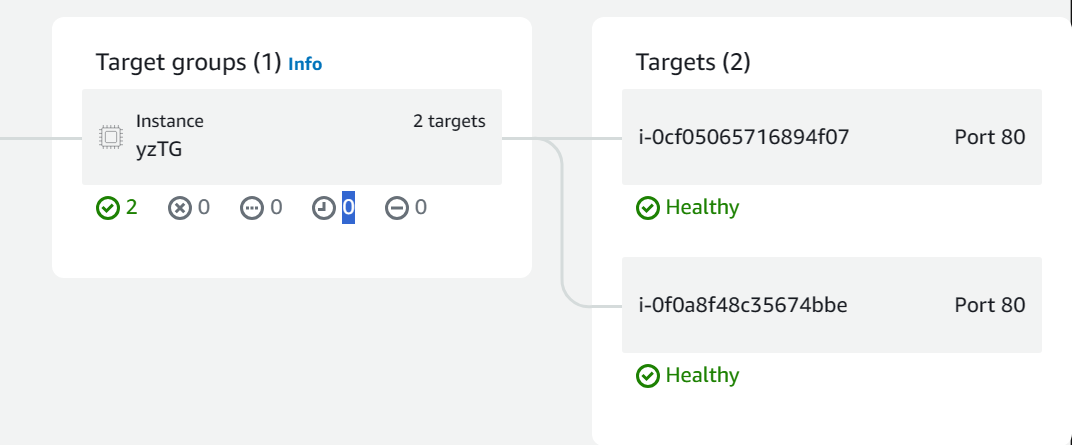


A screenshot of a computer

Description automatically generated

* Now, we are going to create an application load balancer as we did in the WAF lab. Follow the same steps of creating a load balancer, and make sure that you access the website trough a shared domain name.

Provide a screenshot of the health check status



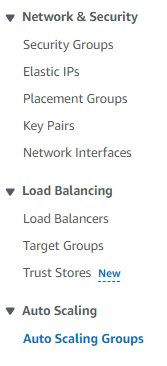
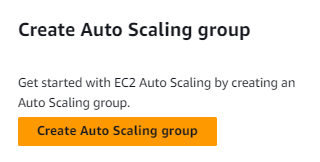
Provide a screenshot of the web page after accessing it from the domain name provided by the ALB

A screenshot of a computer

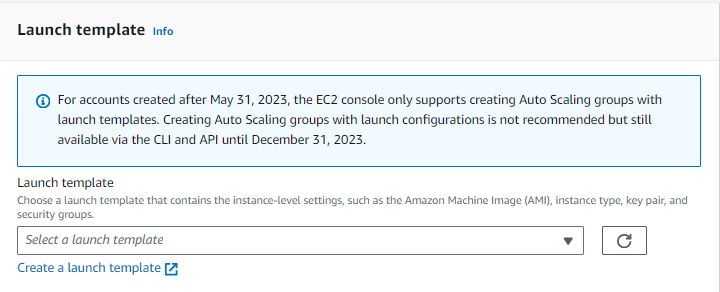
Description automatically generated

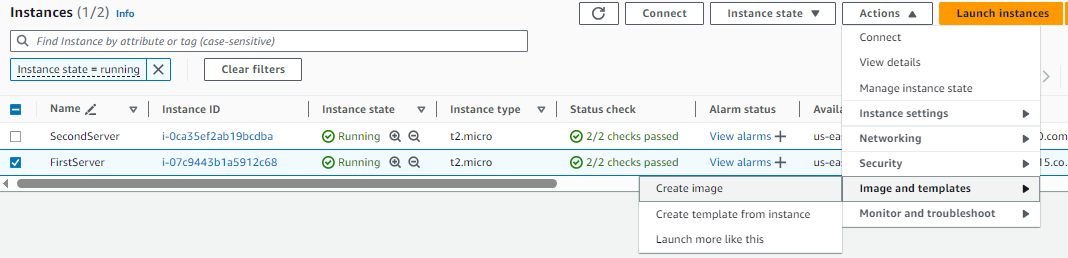
**Part 2: Enabling the auto scaling feature**

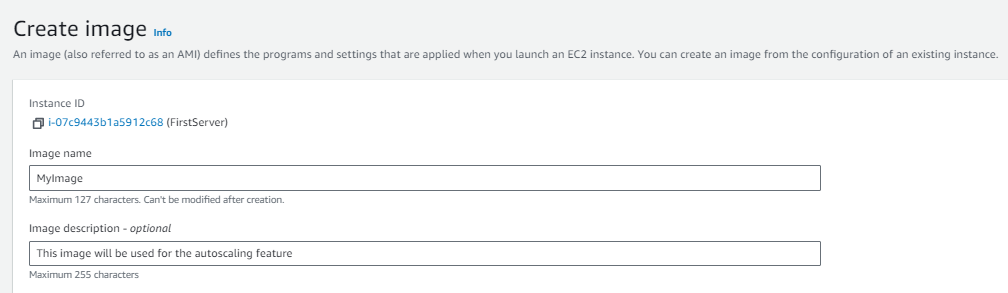
* In this part, we are going to enable the auto scaling feature, which allows the group to scale automatically as the load on your application spikes higher or lower, based on demand.
* Go to **auto scaling groups**, and click on **create auto scaling group**

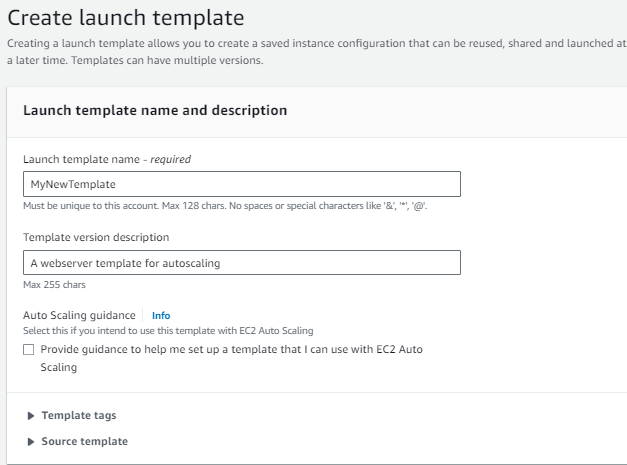
* Put a name for the auto scaling group.
* In Launch template, we have to create a new template to define the configuration settings for the instances that the Auto Scaling group launches.
* Click on **Create a launch template**.



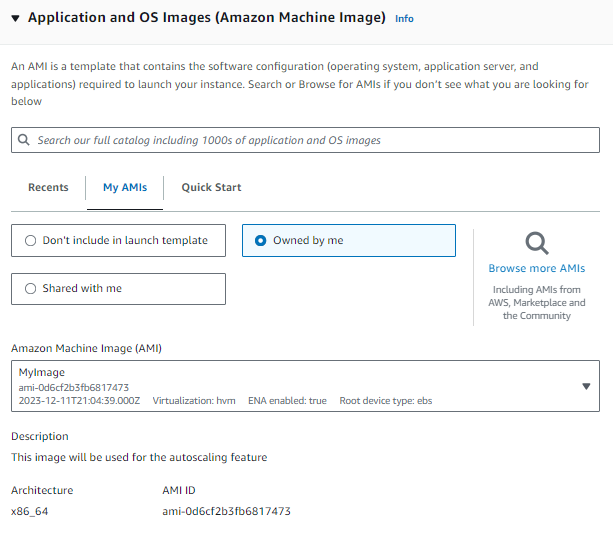
* For the instance launching to be completed, we have to create an Amazon Machine Image (AMI), which is a pre-configured virtual machine image. When auto scaling launches instances based on the template, it uses the specified AMI to create the virtual machines with the defined configuration.
* To create an AMI, go to one of the virtual machines, click on **Image and Templates**, and then select **Create Image**.
* Put a suitable name and description for the image, as follows, and create the image.



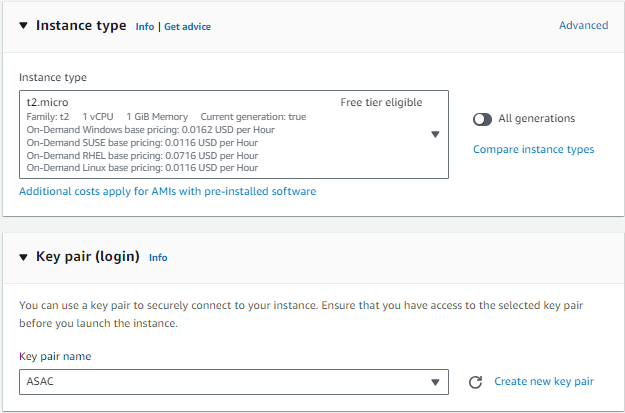
* After creating the image, we can launch the template. Now, put a suitable name and description for the template, as follows.



* After creating the image, we can launch the template. Now, put a suitable name and description for the template, as follows.
* In **application and OS images**, choose the AMI that you have just created.

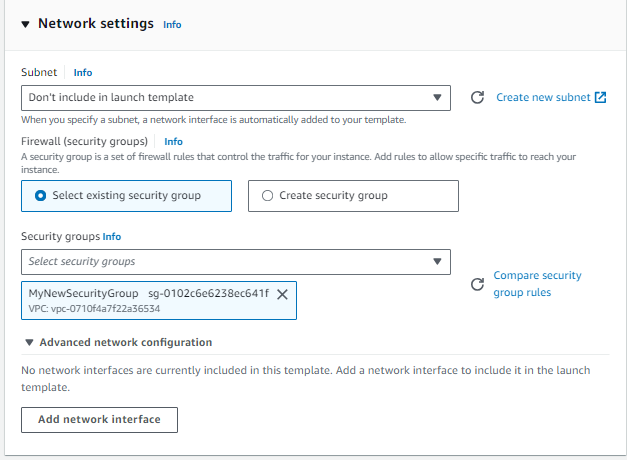


* In **application and OS images**, choose the AMI that you have just created.
* Choose a suitable instance type, and key pair.



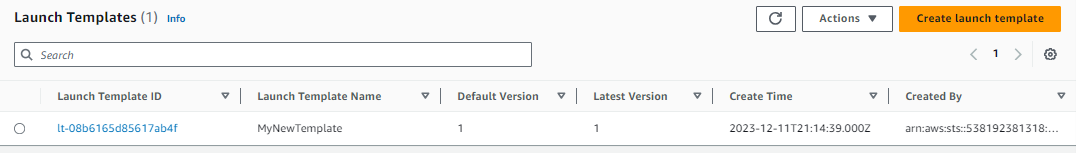
In the network settings, create a security group and define some rules. Within the rules, include the following:

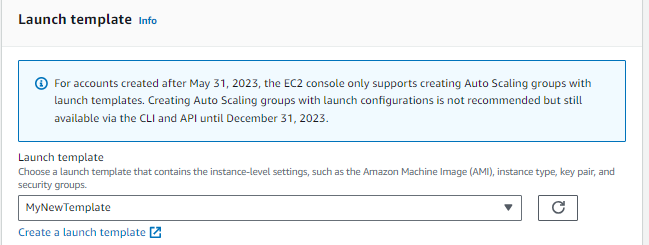
1. Allow HTTP traffic.
2. Restrict SSH connections to machines, allowing access only from a specific IP address (**your machine's IP address**).

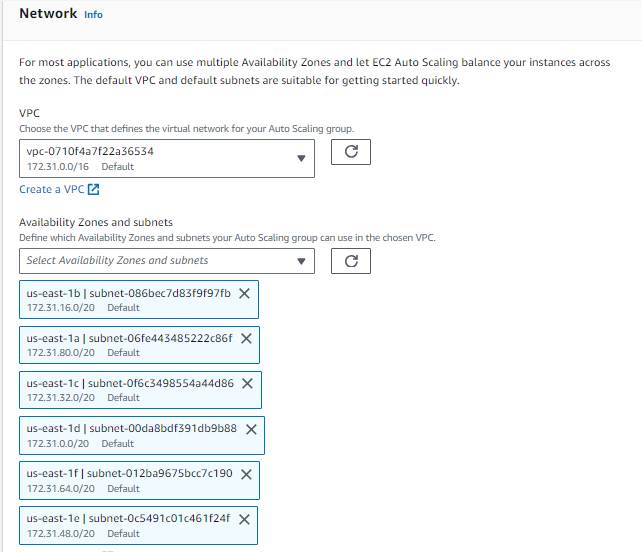


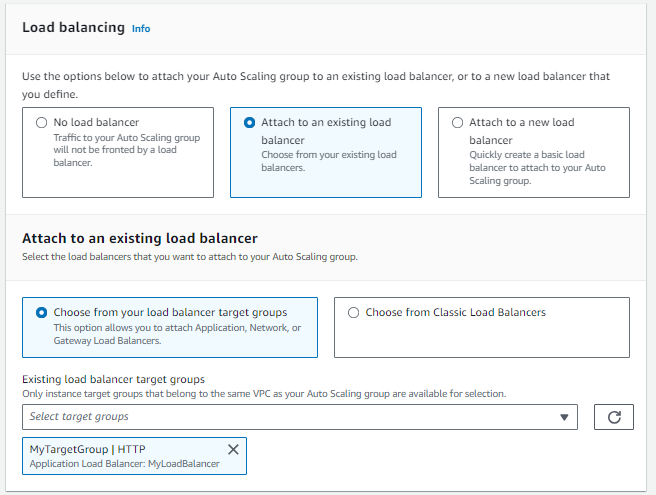
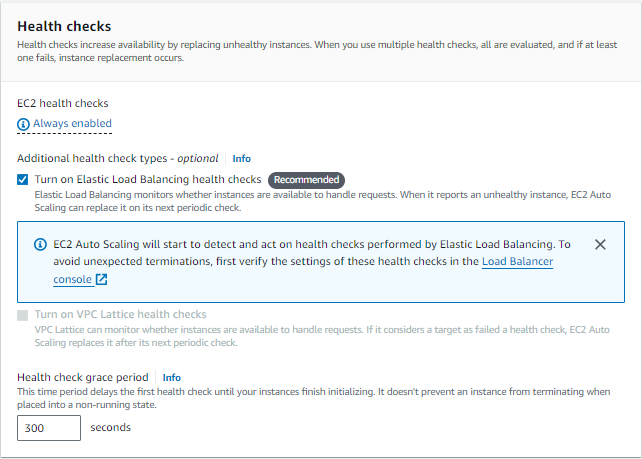
* Finally, click on “**Create lanuch template**”

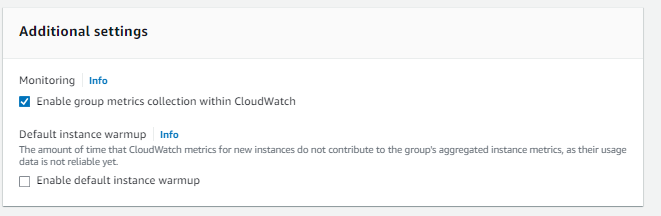




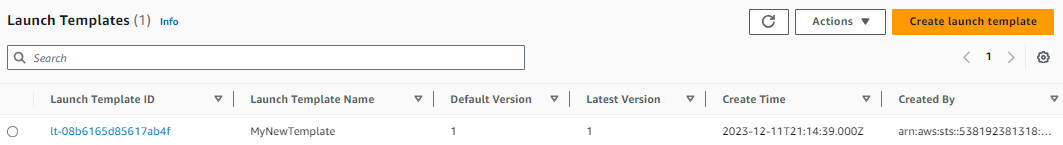
* Now, we will return to **Create Auto Scaling group** and enter the created launch template. Specify the launch template that you have created.
* In the Network part, we are going to choose the VPC and availability zones. We are going to choose all AZs, as follows.



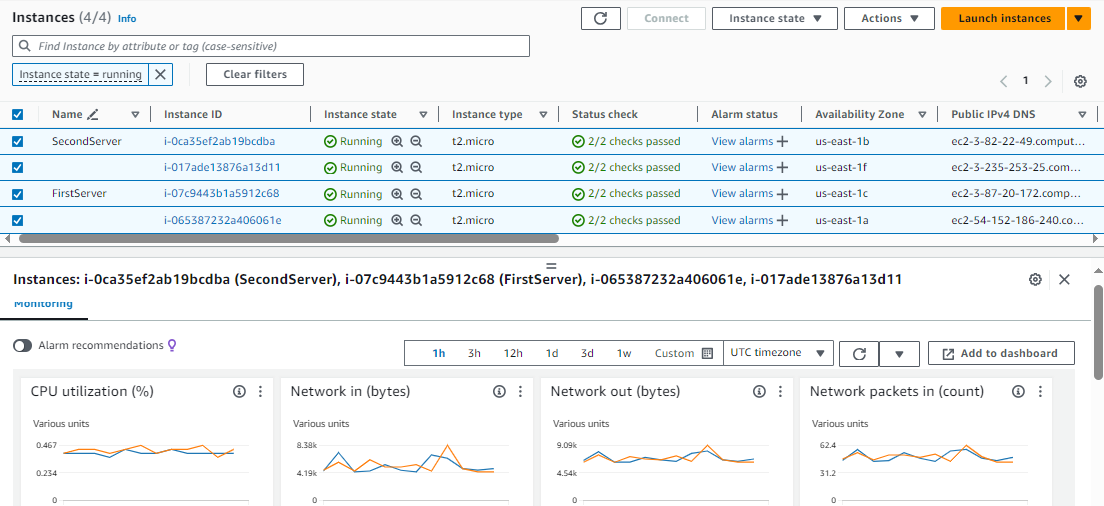
* In the load balancing part, we will attach attach our Auto Scaling group to an existing load balancer (the one that we created before).
* In the load balancing part, we will attach attach our Auto Scaling group to an existing load balancer (the one that we created before).
* In health checks part, turn on Elastic Load Balancing health checks.
* Enable group metrics collection within CloudWatch.



* For Group size and scaling sections, choose the following values:
* Group size – Desired capacity: 2
* Scaling - Scaling limits – Min desired capacity: 2
* Scaling - Scaling limits – Max desired capacity: 6
* In Automatic scaling, we will choose Target tracking scaling policy. For metric type, choose Average network in (bytes). Set the target value to specific value (100 or 1000) and instance warmup to 10 seconds.
* Now click on Create auto scaling group.



* Wait for five minutes and observe the number of EC2 instances created. Explain the results you see on your screen in detail. Additionally, clarify what is meant by autoscaling and why it is necessary. Provide a screenshot.
* Notice the traffic of your instances.



* Explain the results that you see on your screen in detail (Network packets in, CPU utilization, …).

CPU Credit Usage:

CPU credits are a feature of some cloud platforms (like AWS's T2 instances) that allow instances to burst above their baseline CPU capacity.

The graph shows the count of CPU credits consumed over time.

The spikes in CPU credit usage correspond to the spikes in CPU utilization, which makes sense as the instances would use credits to handle the increased load.

CPU Credit Balance:

This represents the number of unused CPU credits that an instance has accumulated over time.

The balance decreases when credits are used, as seen by the downward trend in the graph.

Network Packets In/Out:

These metrics indicate the number of network packets received/sent by the instance.

The 'Network packets out' graph is shown with a sudden drop in packets at around 07:00, which could indicate a decrease in outgoing traffic or a resolution of a task that was generating traffic.

Status Checks:

The '2/2 checks passed' indicates that the instance has passed both system status checks and instance status checks, meaning there are no detected issues with the instance or the system it is running on.

Alarms:

There are options to 'View alarms' which would show any triggered alarms for performance thresholds that may have been set for these instances.

* Provide a screenshot of the traffic nature that you have observed.

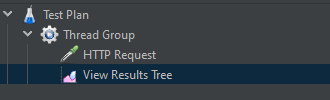
A screenshot of a computer

Description automatically generated

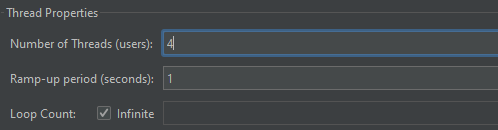
**Part 3: Testing auto scaling features**

To test our configuration, we are going to use the JMeter tool, an open-source software testing tool designed to perform performance, and load testing on applications. Follow the following steps to build your testing plan:

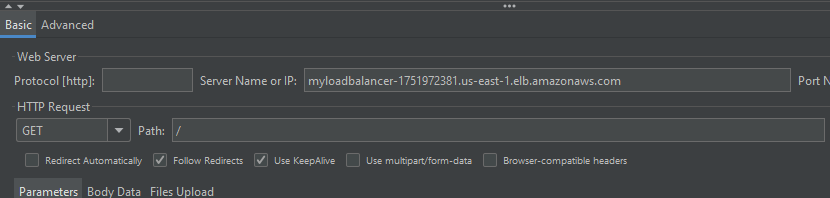
* To download the JMeter you must have the Java 8 . 2. Then download JMeter from Binaries choose the Zip file. ([Download Java for Windows](https://www.java.com/download/ie_manual.jsp))
* From downloaded folders in your device extract the zip file and then copy it to C drive. ([Apache JMeter - Download Apache JMeter](https://jmeter.apache.org/download_jmeter.cgi))
* Now you can launch from the extracted files.
* Upon launching the program, initiate by right-clicking on the test plan, then select Add -> Threads -> Thread Group.
* After that, right click on the threat goup, then select Add -> Sampler -> HTTP request.
* Also, right-click on the threat goup, then select Add -> listener -> View results tree, as follows:



* For thread group, choose the following settings:



* For HTTP request, fill the server name with **your load balancer domain name** and the path with **/,** as follows:



* Now, start your scan and oberve the results on view results tree screen!
* Provide a screenshot of your results on view results tree screen

**A screenshot of a computer

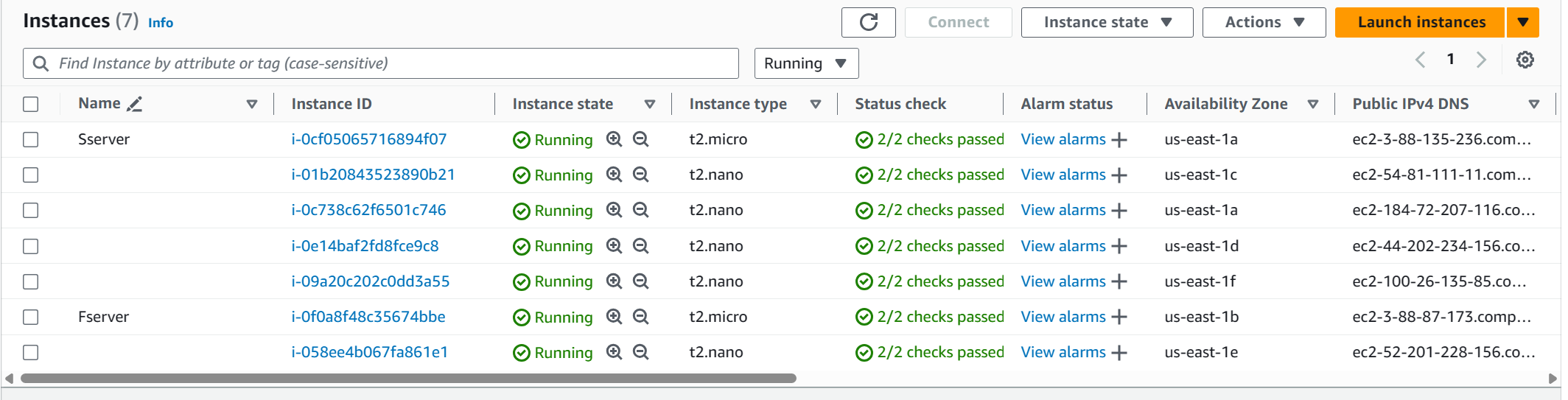
Description automatically generated**

* Now, go the to the see the traffic of your instances over 5 minutes period and observe the difference between the values before and after (Concentrate particularly on Network packets in). Provide a screenshot of your results after 5 minutes.

A screenshot of a graph

Description automatically generated

* Now, its’ the time to observe the auto scaling feature. See the number of inctances after sending a large number of packets from Jmeter to the target. Provide a screenshot of the instances and explain what happened and thre reason behind this.

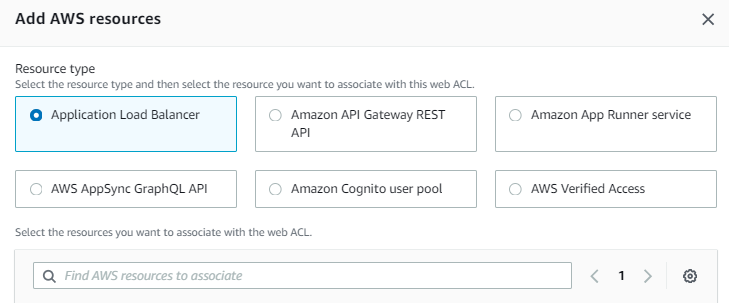


**Configuring security measures**

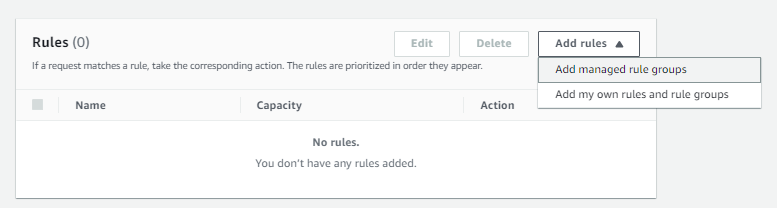
**Part 1: Using managed rules on WAF – Bot Control Prevention**

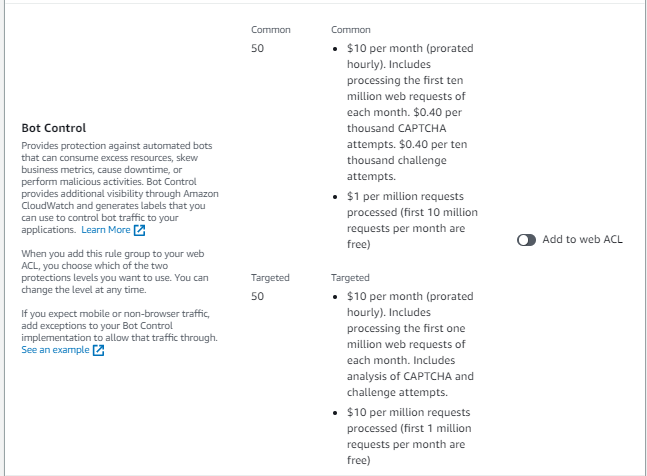
Now, let’s try to use the managed rules at WAF to secure our website from bot control. Bot Control helps you reduce costs associated with scraper, scanner, and crawler web traffic. With Bot Control, you can easily monitor, block, or rate limit bots such as scrapers, scanners, crawlers, status monitors, and search engines.

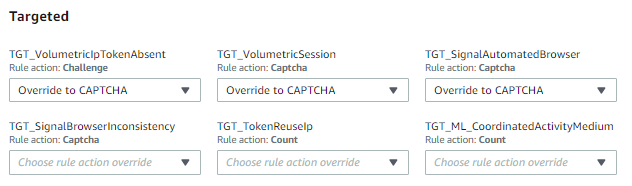
* Go to WAF service. We are going to create a web ACL.
* Go to Web ACLs, and press on “**Create web ACL**”
* Name your ACL.
* Name your ACL. Then, choose “**Application Load Balancer”** from “**add AWS resources**”.



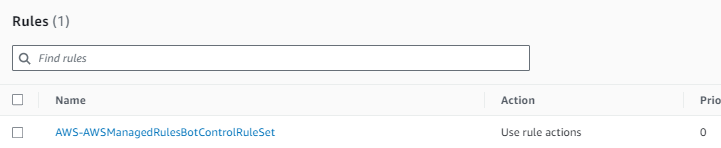
* Search and add your previously created load balancer.
* Next, we have to add some rules to our web ACL. Choose “Add managed rules”.



* From “AWS managed rule groups”, enable “Bot Control”, and click on “Edit”. 
* From “Bot Control inspection level”, choose targeted.
* Override these options to CAPTCHA, and leave other things on default settings, then save the rule.

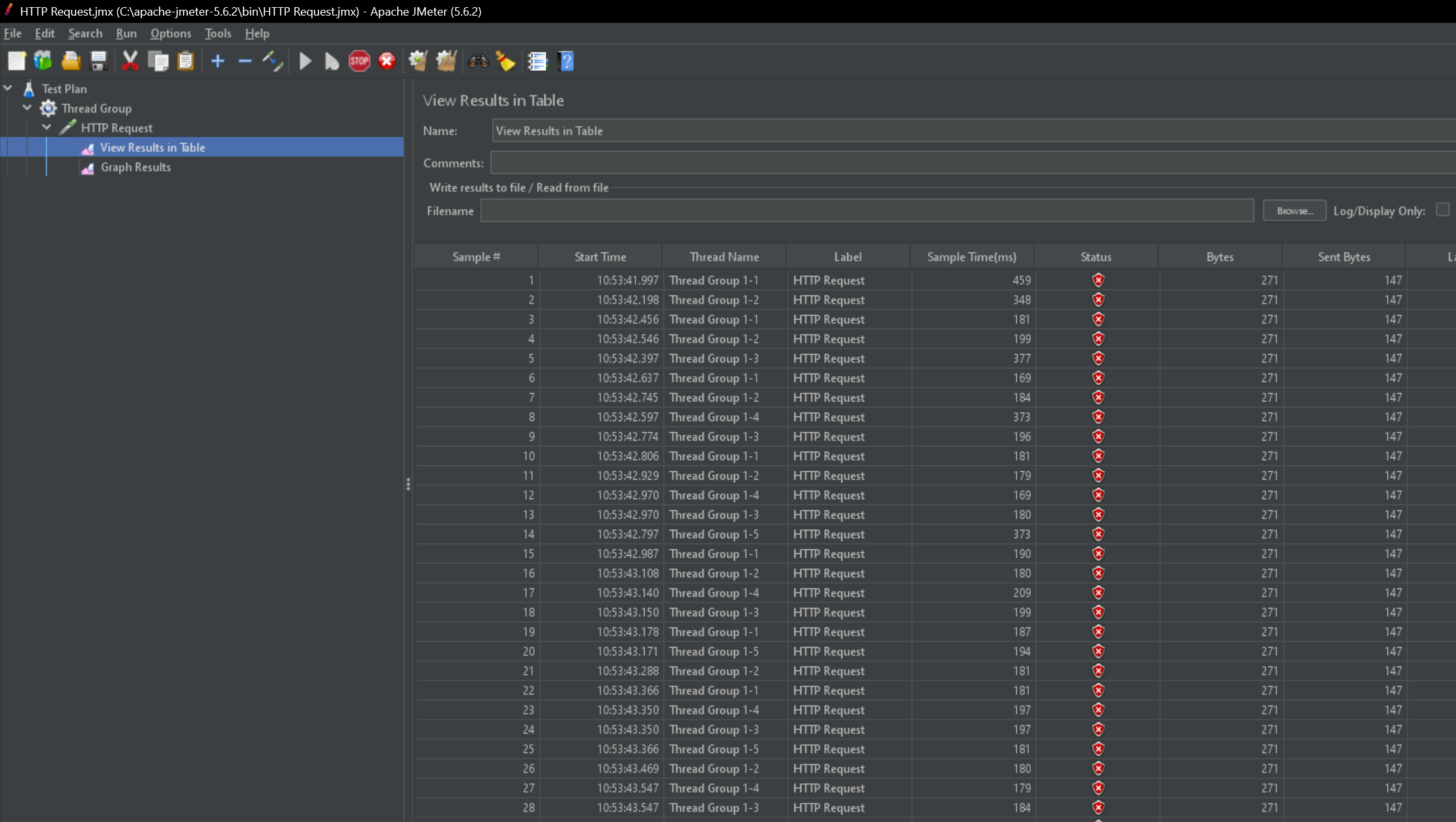


* Now, click on add rules. There should be a one rule as follows:

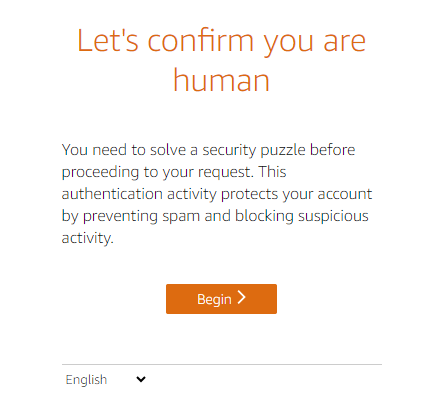


**Part 2: Testing our managed rule on JMeter**

* Let’s re-run our scan again on JMeter and see what happens. Run your scan again and provide screenshot on the “view results trees” section. Additionally, explain precisely what happened and what is the effect of the rule that we have configured on the security of our deployed systems.



* Now, refresh your website (load balancer domain name) and see what happened. You should receive something like this:



* Explain the results that you have seen on your screen.

A screenshot of a computer

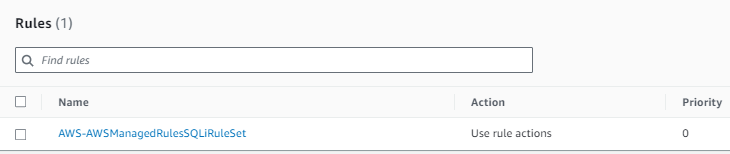
Description automatically generated

**CAPTCHA page, which is a type of challenge-response test used in computing to determine whether or not the user is human. This particular page is likely part of AWS (Amazon Web Services) given the domain name visible in the address bar (amazonaws.com). The page is prompting the user to confirm that they are not a robot by completing a security check. This is a common measure used to protect accounts and prevent automated software (bots) from performing unauthorized actions, such as creating spam accounts or attempting to access user data.**

**Part 3: Using managed rules on WAF – SQL injection and XSS attack prevention**

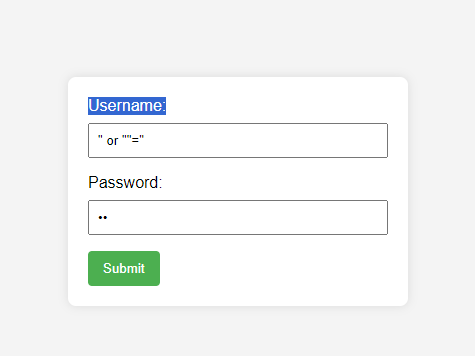
One of the managed rules that exist on AWS allows you to block request patterns associated with SQL injection. This can help prevent remote injection of unauthorized queries. Let’s see how we can manage this.

* Delete the previous rule and create a new managed rule. This time we will choose **SQL database**, as follows:



**Part 4: Testing our managed rule on the website**

* Now, go to the website page and try to enter the following code in the username field:



* Click on submit, and provide a screenshot of the result. Search on the meaning of the HTTP code that you see on the page, and try to figure out there results and the reason behind them.

A screen shot of a computer

Description automatically generated

**screenshot shows a browser page with a "403 Forbidden" HTTP status code displayed. This code indicates that the server understands the request but refuses to authorize it. This might happen if the server's security policy, such as a Web Application Firewall (WAF), detected the input as potentially malicious and blocked the request to prevent an SQL injection attack.**

**If you attempted to submit the login form with the SQL injection patterns in the username, and then received a 403 Forbidden response, it would indicate that the WAF or some other security mechanism on the server successfully detected the attempted attack and blocked it, which is its intended purpose.**

Explore the other managed rules to know more about securing web application.