Using Cloud Armor To Protect Vulnerable Website

Cloud Armor is a Google Cloud tool that offers protection against general and Distributed Denial of Service (DDOS) attacks along with firewall services for web applications.

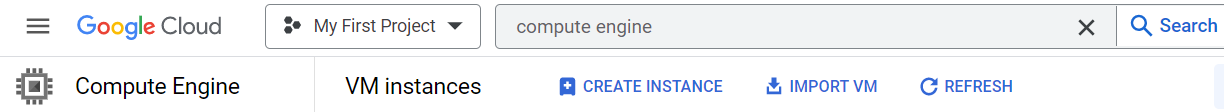
In this tutorial, the OWASP Juice Shop web application shall be used to demonstrate Cloud Armor’s capabilities. The application has been specifically developed with several vulnerabilities for testing purposes.

To begin the tutorial, the web application will first need to be hosted on a GCP Virtual Machine:

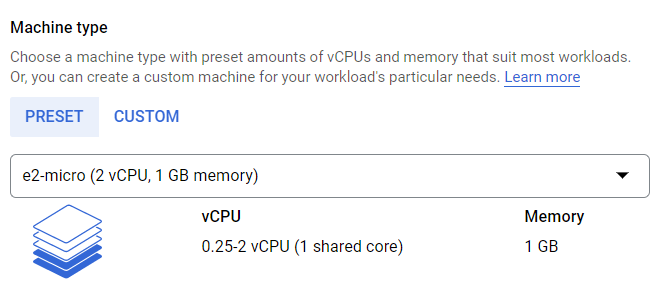
# Step 1: Create a Virtual Machine Instance

To host the web application on a VM, head to **Compute Engine** in the Google Cloud console by simply searching for it in the search bar.

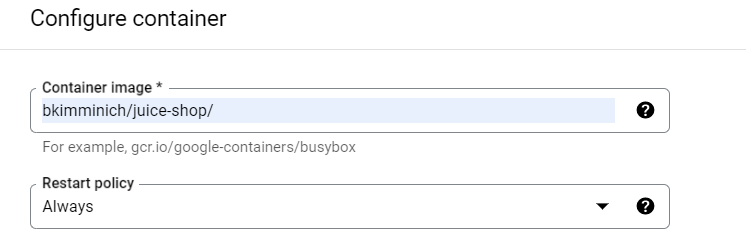
The Compute Engine page will display all created VM instances. Click on **Create Instance** at the top of the bar:



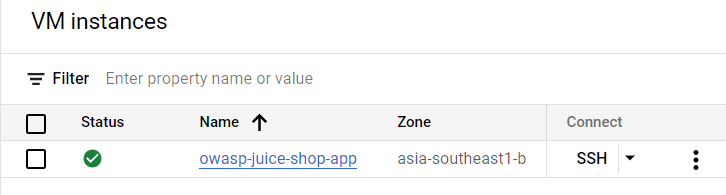
Set the name and regions as per your preference and change the Machine type to the smallest available option - e2 Micro in this case:



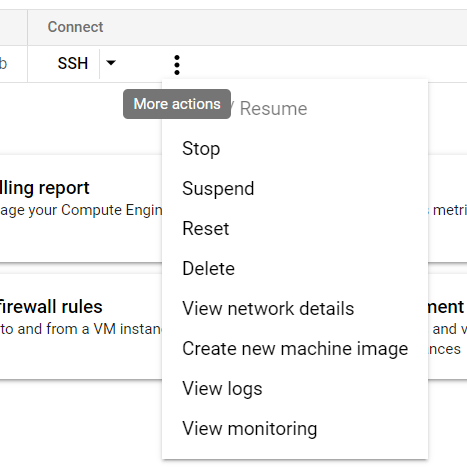
After this, click on **Deploy Container** to add the OWASP Juice Shop App to this specific VM. This is done by typing in bkimminich/juice-shop in the **Container image** field:



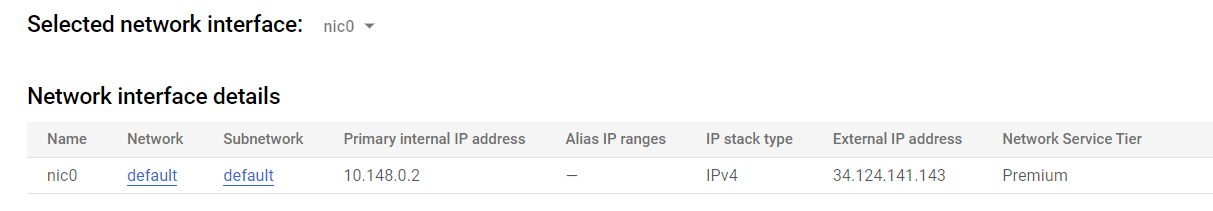
Simply click on **Select** without making any other changes. Scroll down and click on **Create** to create the VM Instance; called owasp-juice-shop-app in this case:



The three dots toggle the VM instance and help reveal the network details of the VM.



The network details include the external IP address of the VM:

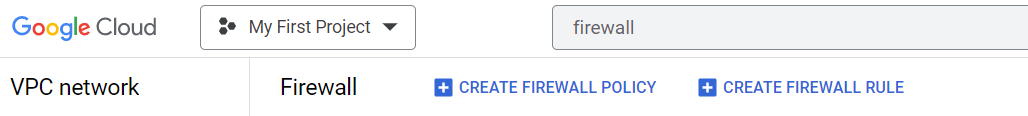


However, to access the web app through this IP address, the firewall rules must be configured through Google Cloud.

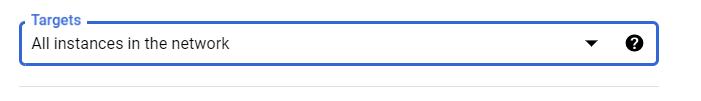
# Step 2: Create a Firewall Rule for the Instance

Search **Firewall** in the search bar to access the Firewall option in Google Cloud’s **VPC Network** product.

Click on **Create Firewall Rule** located in the bar at the top.

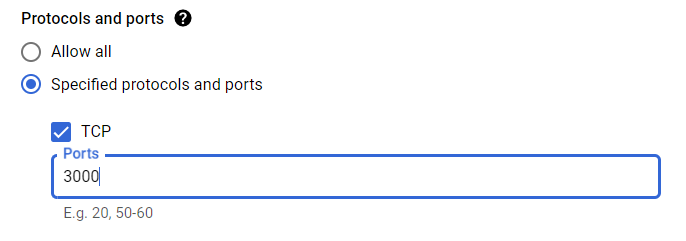


Add a name to the new firewall rule and scroll down to **Targets** option to select **All instances in the network** option from the dropdown menu:



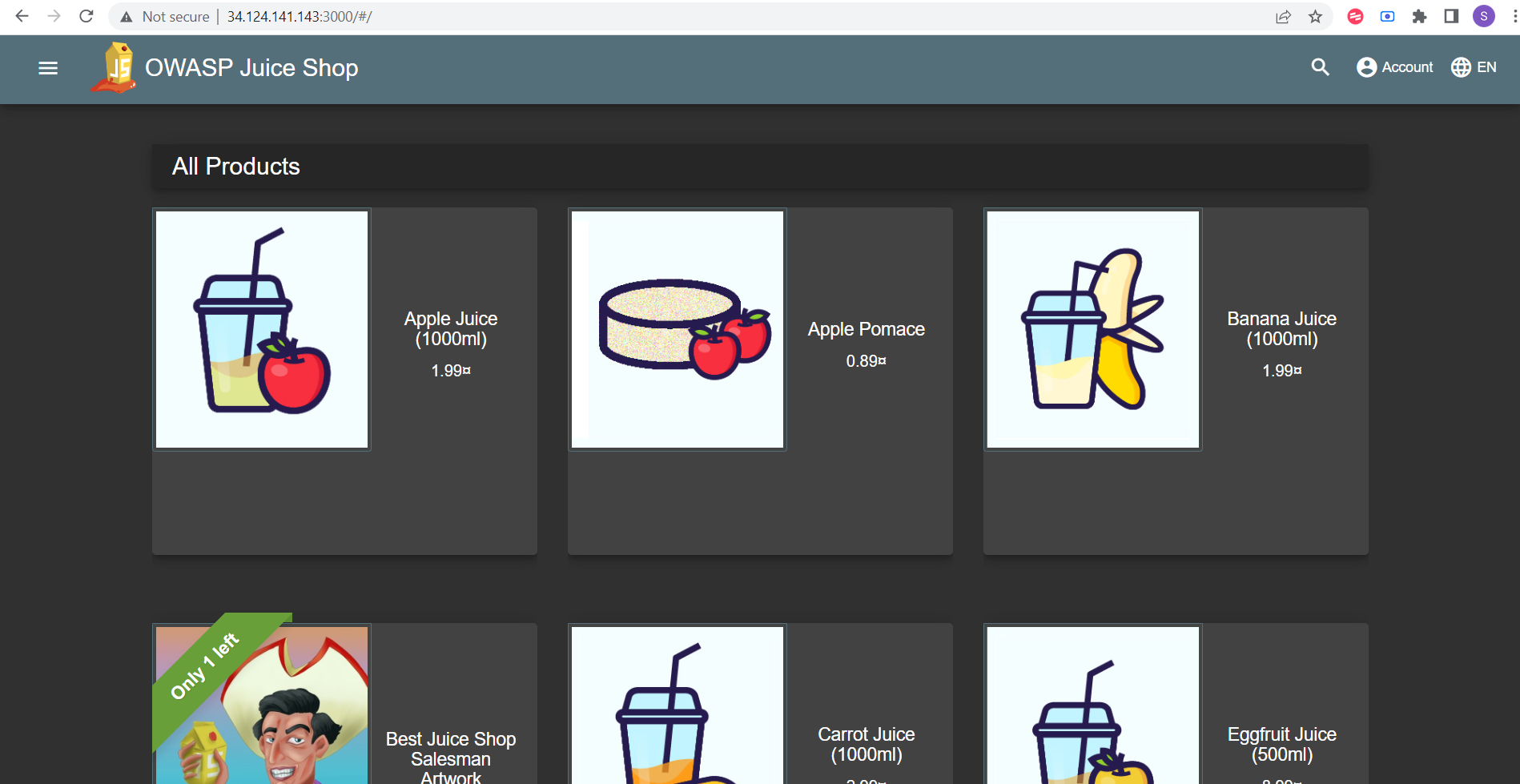
In the **Source IP ranges** option, add **0.0.0.0/0.**

Then, scroll down to the **Protocols and Ports** menu and click on the **TCP** checkbox. Add a port number to access the web application on, such as:



Click on **Create** to launch the firewall rule. The changes will take a few minutes to take effect.

The web app can now be opened at the given IP address by adding the specified port:



However, the web application can not be used in its current state without a load balancer.

# Alternative: Use Google Cloud Shell Commands

Before we get to the load balancer, it is important to see how both of the previous steps can be handled via the Google Cloud Shell.

To create a VM with OWASP Juice Shop and provide it with the requisite firewall rule, simply run the following commands in the Cloud Shell:

gcloud compute instances create-with-container owasp-juice-shop-app --container-image bkimminich/juice-shop

And

gcloud compute firewall-rules create juice-rule --allow tcp:3000

For the virtual machine instance and firewall access respectively.

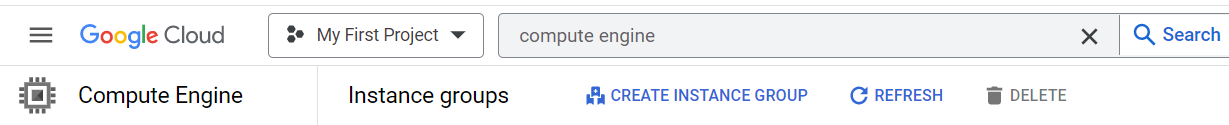
However, it is difficult to specify the region and zone of the virtual machine using this method. Creating a VM instance with the UI allows for far more control over what is executed.

# Step 3: Create an Instance Group for Your Load Balancer

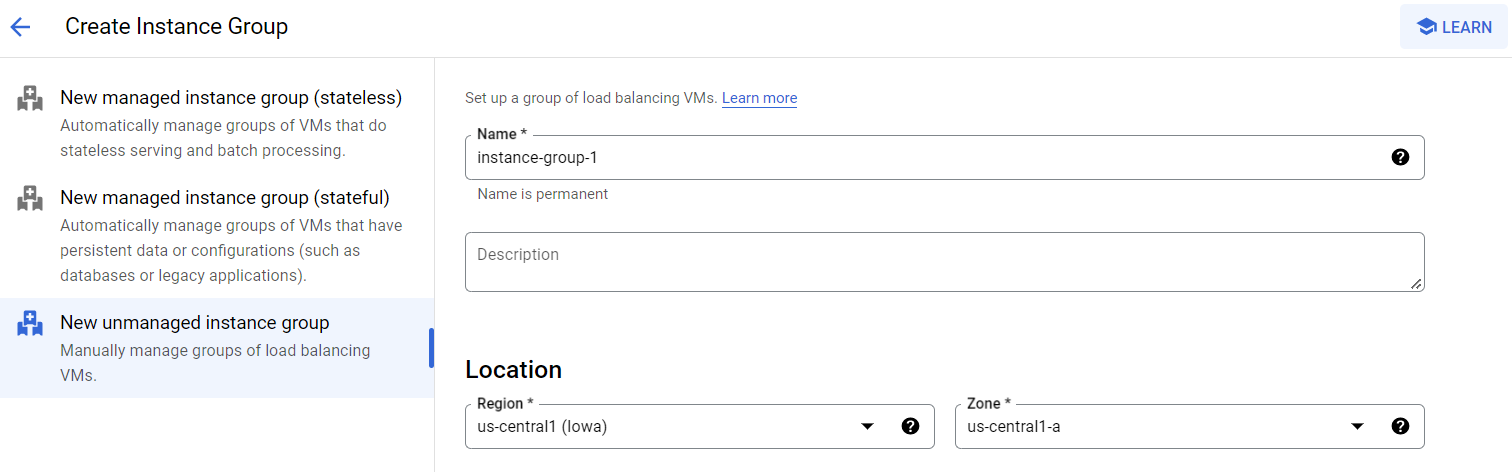
To launch a load balancer for the backend service of OWASP Juice Shop, creating an **Instance Group** to add the created VM instance to is imperative.

For this, simply look for the **Instance groups** option in the **Compute Engine** product menu.

Click on the **Create Instance Group** option located in the bar at the top:



Select the **New unmanaged instance group** from the available options:



Set the name for your new group and change **Region** and **Zone** according to the ones specified in your **VM Instance.** Scroll down to the **VM instances** option and select the VM created earlier.

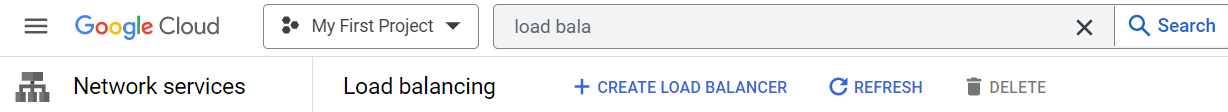
Finally, click on **Add Port** in the **Port mapping** section and specify the **Port Name** as **http** and **Port Number** as **3000 (or whatever number you selected for the firewall rule).**

Finally, click **Create** to make this instance group for your VM instance.

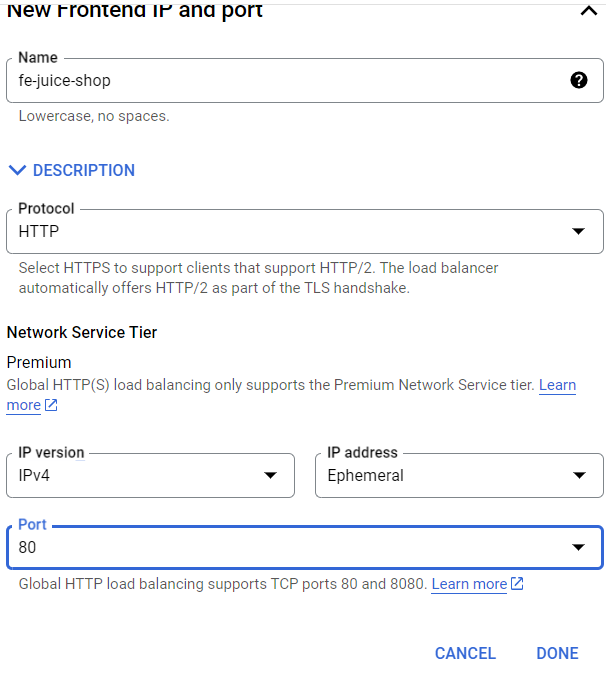
# **Step 4: Create a Load Balancer**

Search for **Load Balancing** in the search bar to find the load balancer management option as part of the **Network Services** of Google Cloud.

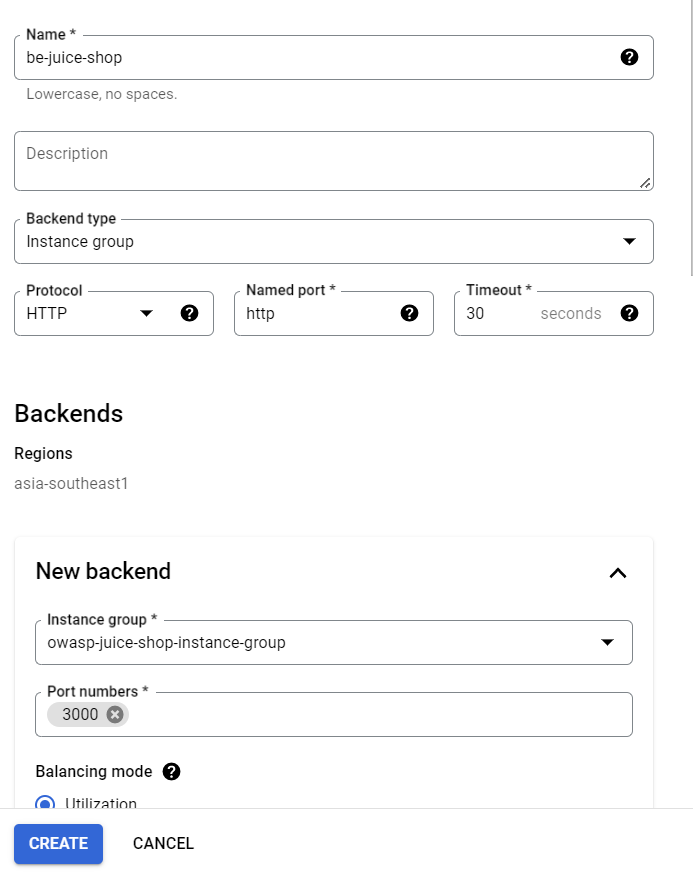
Click on **Create Load Balancer** in the bar at the top:



Give your load balancer an appropriate name and then and do the same for the **Frontend Configuration,** while setting the Port at **80.**

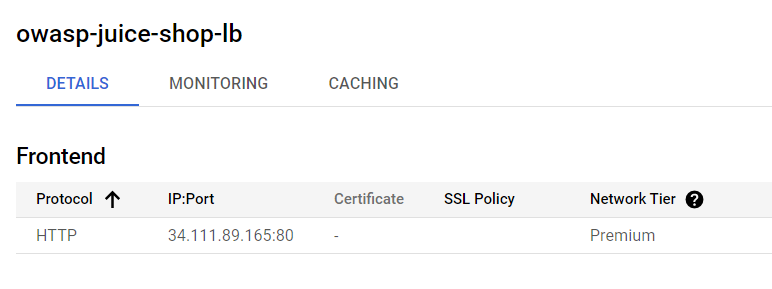


For **Backend Configuration,** click on the available field and select **Create a Backend Service.** Add an appropriate name to the new backend service and choose the created **Instance Group** to initialize the load balancer and select port as specified previously in the instance group (3000 in this case).



Keep everything else as it is and click **Create** to add the backend service. Select this backend service and click **Create** in the load balancer menu to formally create and launch the load balancer.

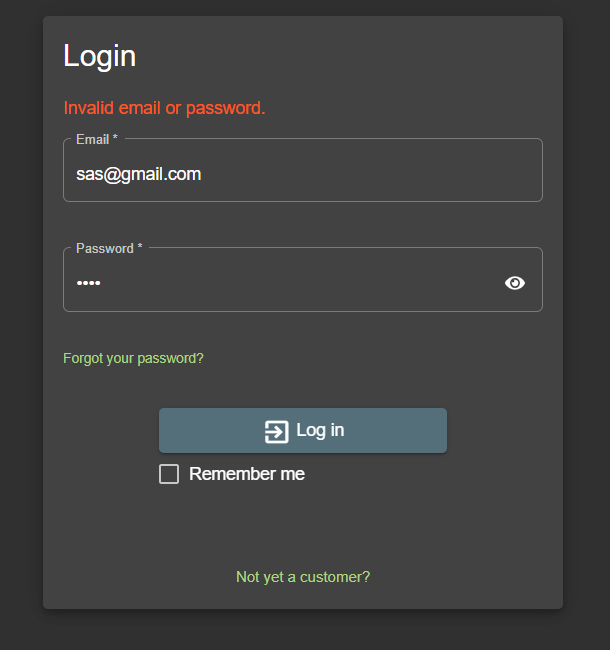
The created load balancer will feature its own IP address which is to be used for accessing the OWASP Juice Shop web app and perform operations on it.



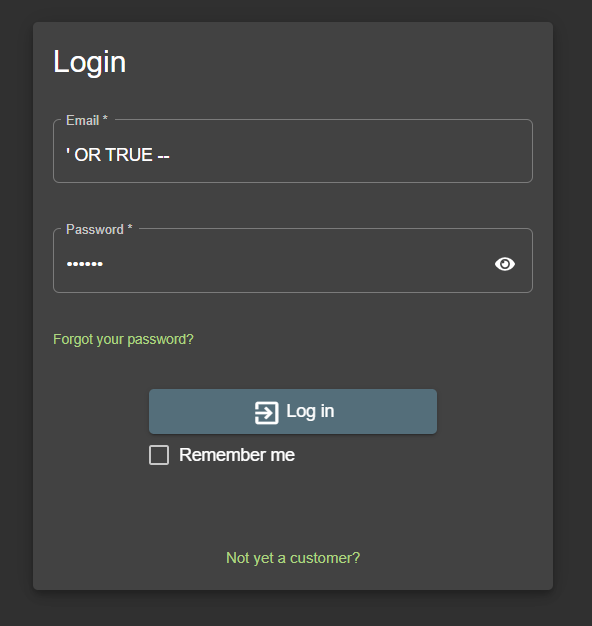
# Step 5: Test Out OWASP Juice Shop

The OWASP Juice Shop web application features several vulnerabilities for developers to test out. By design, the Google Cloud load balancer protects the backend service against **Layer 3 and Layer 4** DDOS attacks. However, other attacks such as the SQL injection are not prevented by the load balancer.

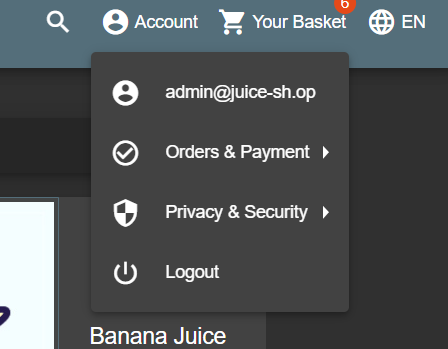
To test this out, try adding a random email and password combination to the login form (accessed by clicking on account) on the Juice Shop webpage. It will reject the credentials after performing a simple form check:



However, an SQL injection statement will simply log in to the Juice Shop as an admin because it gets the very first account from the available list:



And:

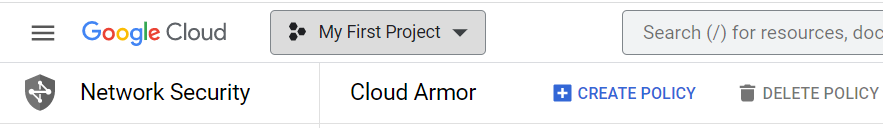


To stop this SQL injection attack, the web application and the overall VM instance needs to be protected with Cloud Armor.

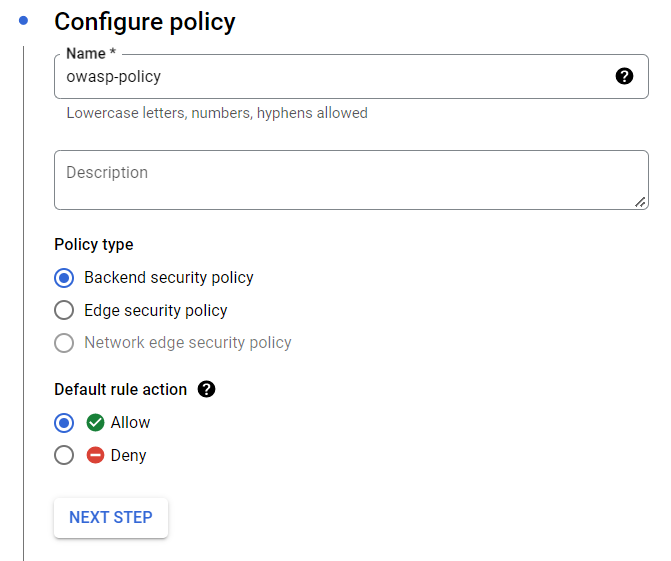
# Step 6: Setting Up Cloud Armor

Now, to protect the web application against SQL injection and other attacks, set up Cloud Armor. Cloud Armor is an enterprise grade product that provides protection to web applications and can help prevent layer 7 attacks as well.

To start off, simply search for Cloud Armor in the search bar and open up the Cloud Armor product page. There, you need to click on **Create New Policy:**



Give your new policy and appropriate name and keep the **Policy type** as **Backend Policy.** The default rule for any new policy is to deny access. However, the default rule for the OWASP Juice Shop is to allow any IP address to come through to the application. Change the **Default rule action** to **Allow** for this.



Click on **Next Step** to set up the very first rule. Since the attack being prevented is an SQL injection, knowing which policy needs to be prevented is important. The list of requests that can be prevented by Cloud Armor can be accessed via the Google Cloud Shell. Simply open the shell and enter the following command:

gcloud compute security-policies list-preconfigured-expression-sets | more

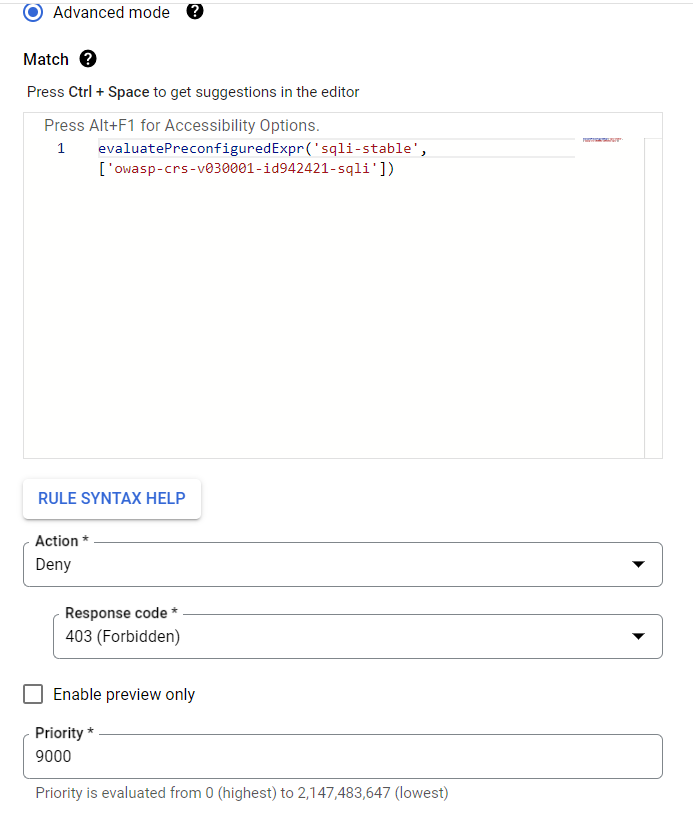
The list of rules can also be found [here.](https://cloud.google.com/armor/docs/waf-rules)

The expression sets of interest are the sql ones. To prevent the SQL injection, configure the new rule by adding an appropriate name and then clicking on **Advanced Mode** to open up a text box where code can be added in CEL language. The code to be added in this particular case is:

evaluatePreconfiguredExpr(‘sqli-stable’, [‘owasp-crs-v030001-id942421-sqli’])

Choose **Action** as **Deny** and the **Response code** as **403.** The **Enable Preview Mode** launches the rule but limits the rule to logging attacks rather than preventing them on the web application. Therefore, leave it unchecked.

The **Priority** of the rule defines the precedence given to every single rule added through Cloud Armor. A higher number results in lower priority, with the maximum number being over 2 million. Add an appropriate number to the **Priority,** such as 9000:



Click on Next to reach the **Add Policy to Targets** section. There, simply choose the **Load balancer backend service** in type one and select the target as the load balancer launched earlier. Do nothing else and click on **Create** to launch the policy.

The policy will take a few minutes to take action. Test the OWASP Juice Shop web application with the same SQL injection again to see if it works. The effective policy should prevent the injection and send this as an output:

