**Aim** :- Implement CloudFront for caching and application security.

**Lab overview and objectives**

In this lab, you will create an Amazon CloudFront distribution to reduce network latency for café website users and deliver the website securely over HTTPS. You will also secure access to the website and REST API endpoints using AWS WAF, which is a service that provides a web application firewall. Finally, you will configure a CloudFront function on the website and adjust the cached file expiration time on the website content.

After completing this lab, you should be able to:

* Create a CloudFront distribution to cache Amazon Simple Storage Service (Amazon S3) objects
* Configure a website hosted on Amazon S3 to be available through HTTPS using CloudFront
* Secure access to the CloudFront distribution based on the network origin of the request
* Secure a REST API endpoint based on the network origin of the request using AWS WAF
* Configure a CloudFront function to affect website behavior from the edge
* Adjust max-age caching settings on a CloudFront distribution

**AWS service restrictions**

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

**Scenario**

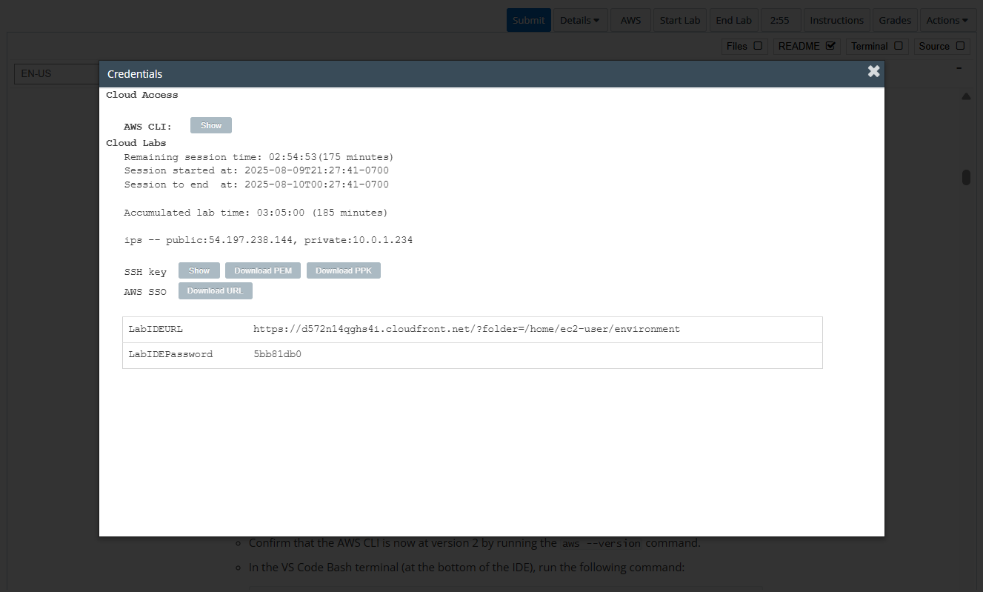
Sofía is pleased with how the café website development project is coming along. She has developed the core serverless application that displays menu items on the website. She also integrated the coffee suppliers web application into the main site and is using Amazon ElastiCache features for the suppliers part of the site.

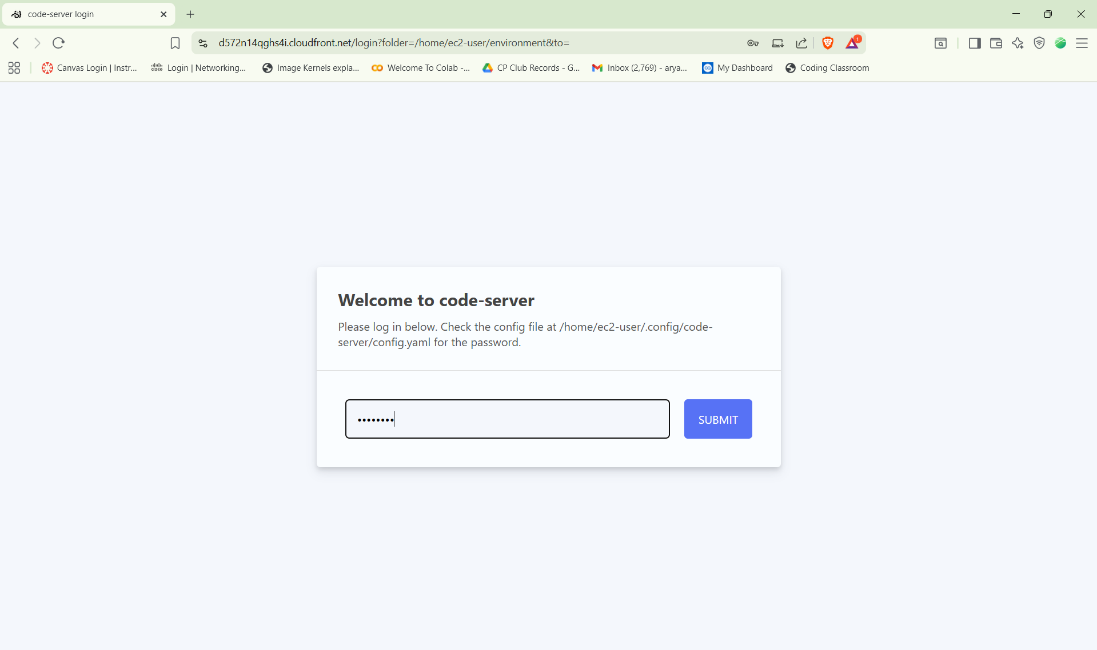
However, she knows that some essential features are still missing. One feature is that the website still runs on HTTP and does not yet support HTTPS. Sofía also wants to ensure that the website will load quickly for users globally. She knows that AWS has many Regions and Availability Zones, but they also have edge locations, which are even closer to users around the globe. She decides to host the café website on a proper content delivery network (CDN), and she has opted to use the CloudFront service.

In this lab, you will again play the role of Sofía to continue to develop the café's web application.

**Task 1: Preparing the lab**

Connect to the VS Code IDE.

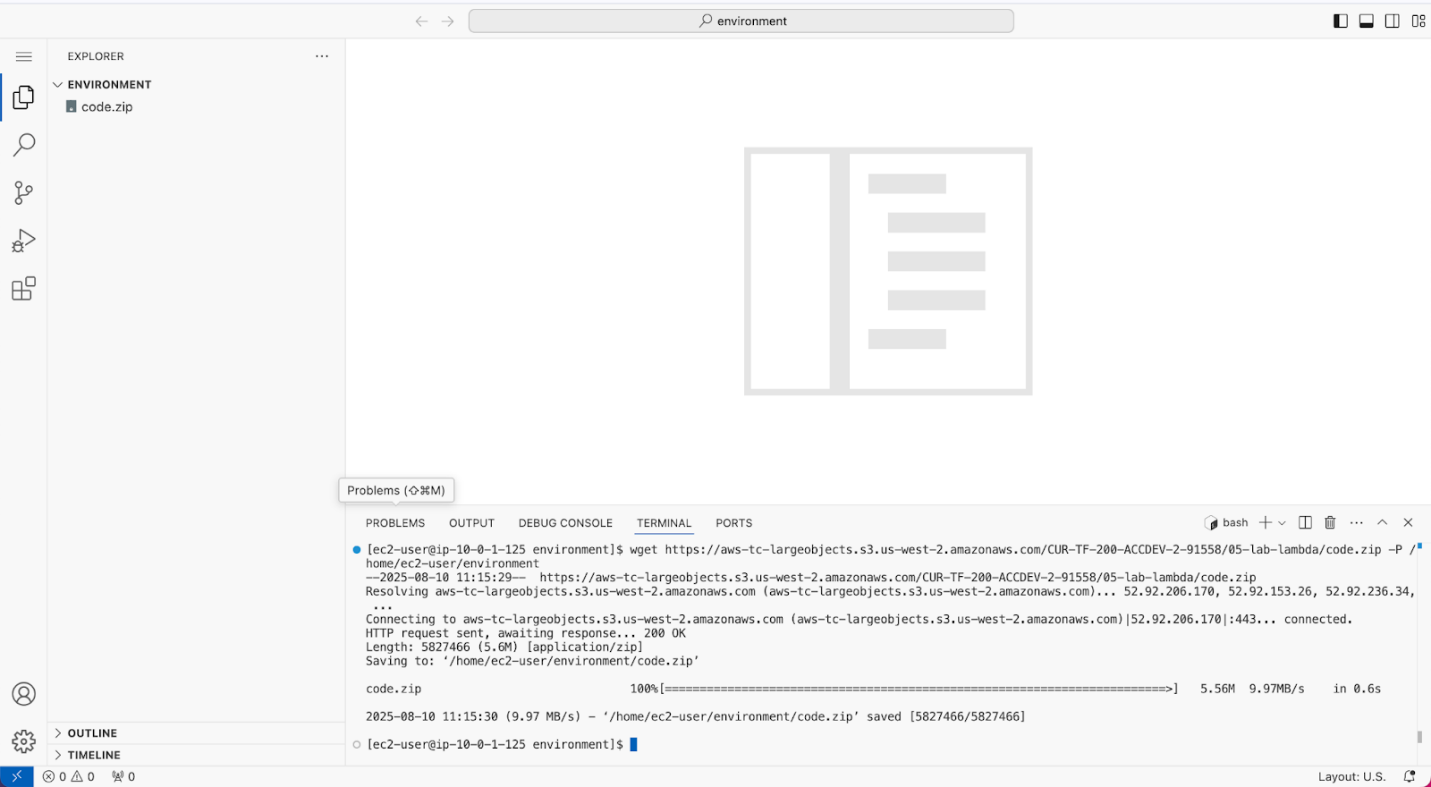
1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   1. **LabIDEURL**
   2. **LabIDEPassword**
3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.

****

1. Download and extract the files that you need for this lab.

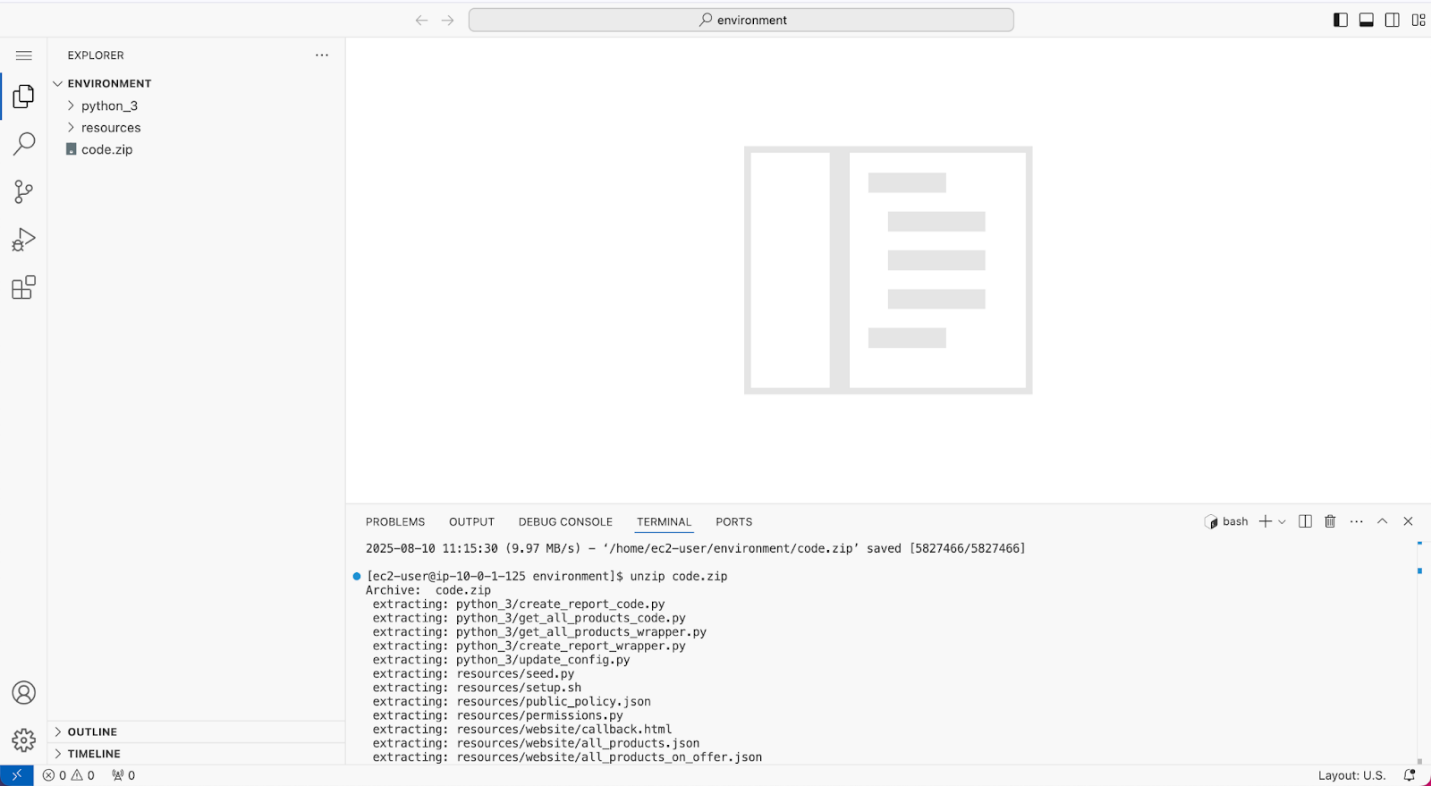
* In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

**wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/09-lab-cloudfront/code.zip -P /home/ec2-user/environment**



1. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

* Extract the file by running the following command:

unzip code.zip

1. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.

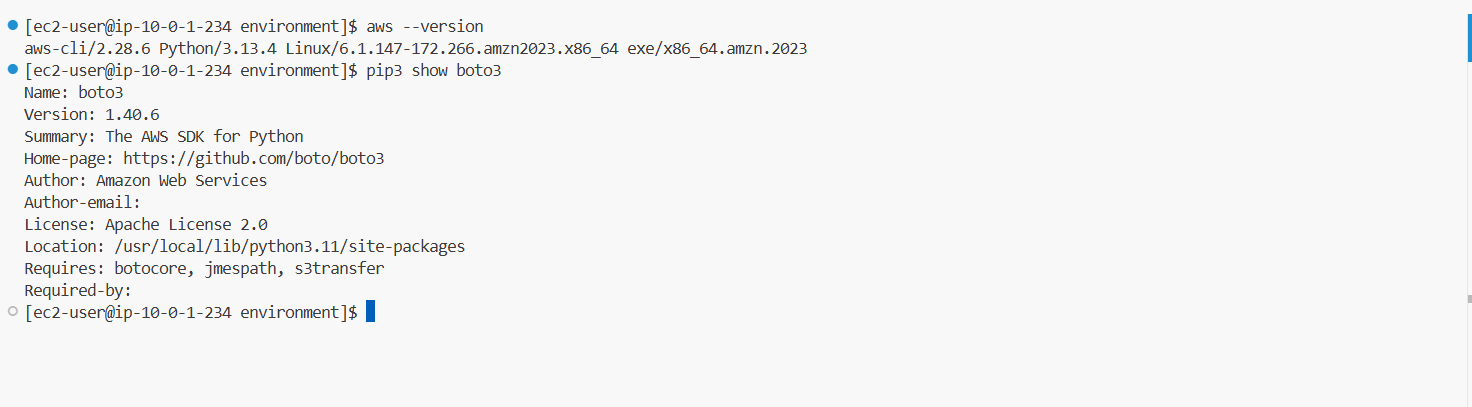
* To set permissions on the script and then run it, run the following commands in the Bash terminal:

chmod +x ./resources/setup.sh && ./resources/setup.sh

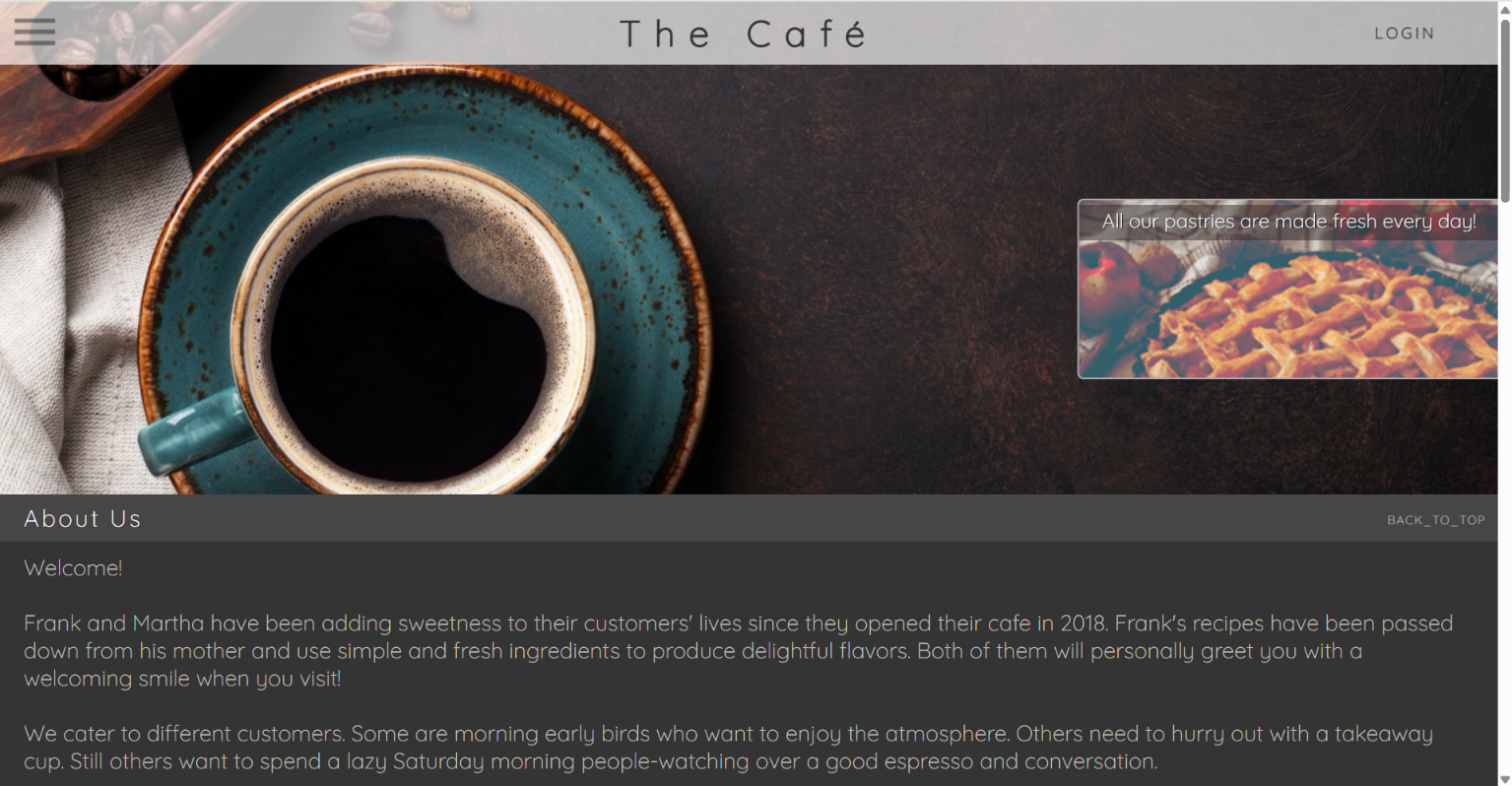
When prompted for an IP address, enter the IPv4 address that the internet uses to contact your computer. You can find your IPv4 address at [https://whatismyipaddress.com](https://whatismyipaddress.com/).

1. Verify the AWS CLI version and also verify that the SDK for Python is installed.

* Confirm that the AWS CLI is now at version 2 by running the **aws --version** command.
* In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

**pip3 show boto3**

1. Note the metadata settings on the objects stored in the S3 bucket, and then verify that you can access the café website.

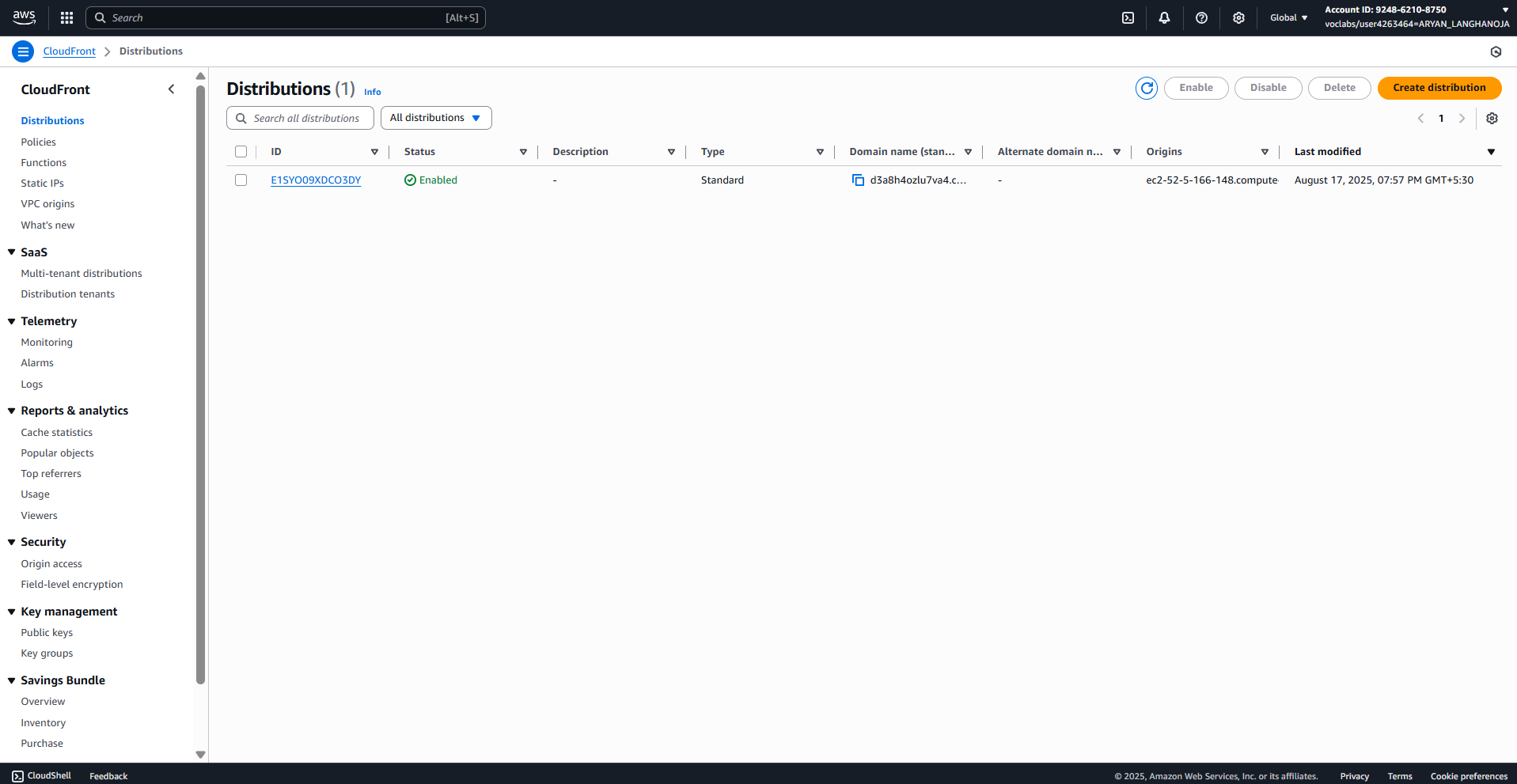
* Navigate to the Amazon S3 console.
* Choose the link for the bucket that has *-s3bucket* in the name.
* Choose the **index.html** link.
* Scroll down to the **Metadata** section.
* Two key-value pairs are listed. The **Cache-Control** key-value pair has a value of **max-age=0**. This was set when you ran *setup.sh* in an earlier step. Line 21 of that script ran the command aws s3 cp ./resources/website s3://$bucket/ --recursive --cache-control "max-age=0" to set this metadata value on every file that it uploaded to the bucket. You'll learn about the significance of this setting later in the lab.
* At the top of the page, open the **Object URL** in a new browser tab.
* The café website displays. If it doesn't, see the following troubleshooting tip.

**Task 2: Configuring a distribution for static website content**

In this task, you will configure the café website, which is hosted on Amazon S3, to be available through a CloudFront distribution. With the distribution, you can enable HTTPS access to the website.

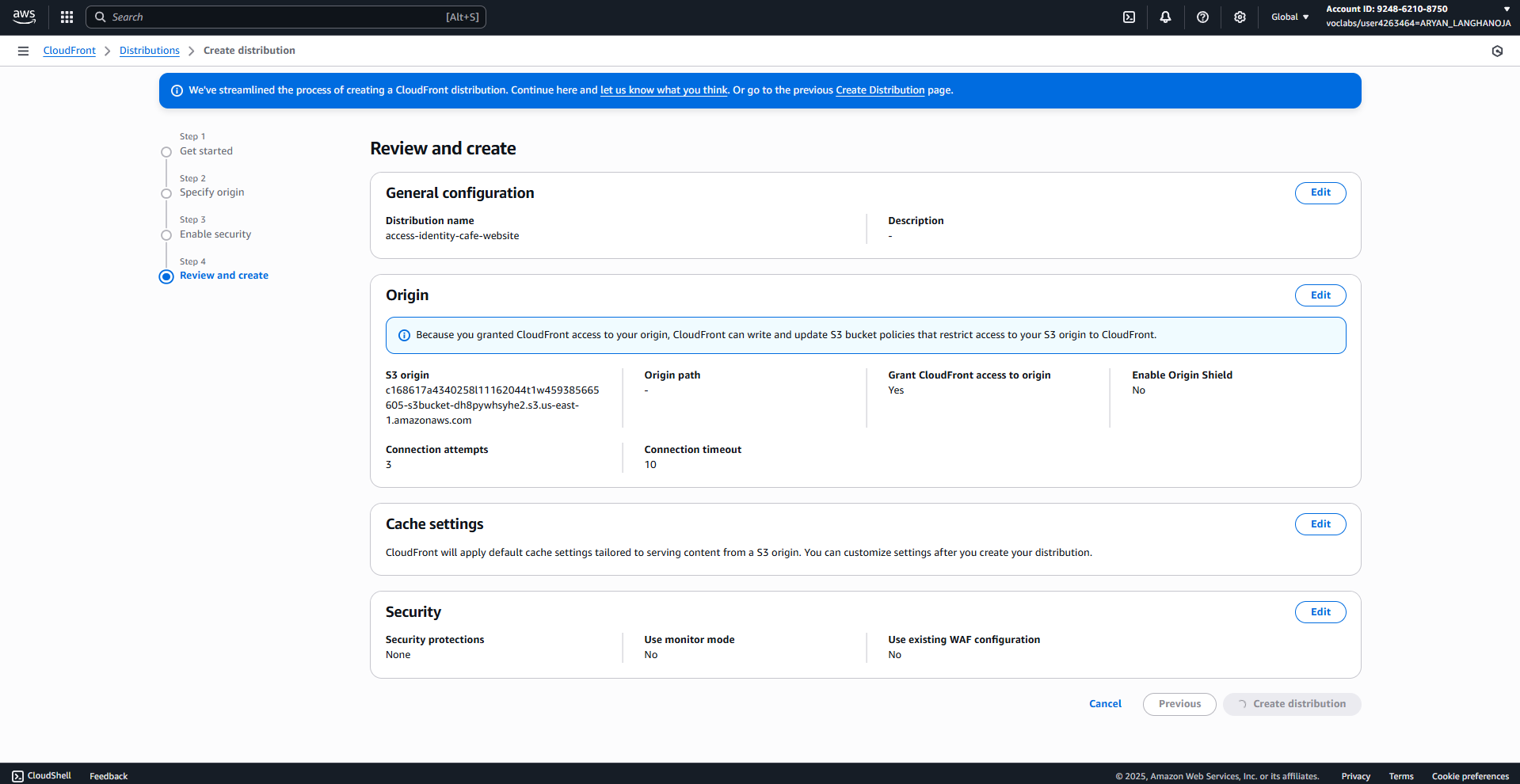
1. Begin to configure a CloudFront distribution for the café website hosted on Amazon S3.

* Navigate to the CloudFront console.
* Choose **Create a CloudFront distribution**.



1. Configure the **Origin** settings for the distribution.

* **Note**: If a setting is not specified in the following steps, use the default.
* **Origin domain**: Search for and choose the S3 bucket that has *-s3bucket* in the name. This bucket contains the website code.
* In the **Origin access** settings.
* Choose **Legacy access identities**.
* Choose **Create new OAI**.
* **Name**: Enter access-identity-cafe-website
* Choose **Create**.
* **Bucket policy**: Choose **Yes, update the bucket policy**.
* In the **Add custom header** section, choose **Add header** and configure:
* **Header name**: Enter cf
* **Value**: Enter 1
* **Note**: The custom header is not an important requirement for this lab, but it will be useful in a later lab.

****

1. Configure the **Default cache behavior** settings.
   1. **Path pattern**: Keep the **Default (\*)** setting. This means that all requests will go to the origin.
   2. **Viewer protocol policy**: Choose **Redirect HTTP to HTTPS**. This will help users find the website, even if they load the HTTP URL.
   3. **Allowed HTTP methods**: Keep the default **GET, HEAD** setting.
   4. **Restrict viewer access**: Keep the default **No** setting. This will be a public website.

1. Configure the **Cache key and origin requests** settings.
   1. Choose **Legacy cache settings**.
   2. Keep the settings for **Headers**, **Query strings**, and **Cookies** as **None**.
   3. **Object caching**: Choose **Use origin cache headers**.

**Note**: Recall the metadata key-value pair with key **Cache-Control** and value **max-age=0**. This metadata is set on all of the objects stored in the origin S3 bucket. By choosing **Use origin cache headers** for this distribution, you ensure that this distribution will inherit this setting applied to all of the objects in the Amazon S3 origin.

* 1. Under **Web Application Firewall (WAF)** choose **Do not enable security protections**.

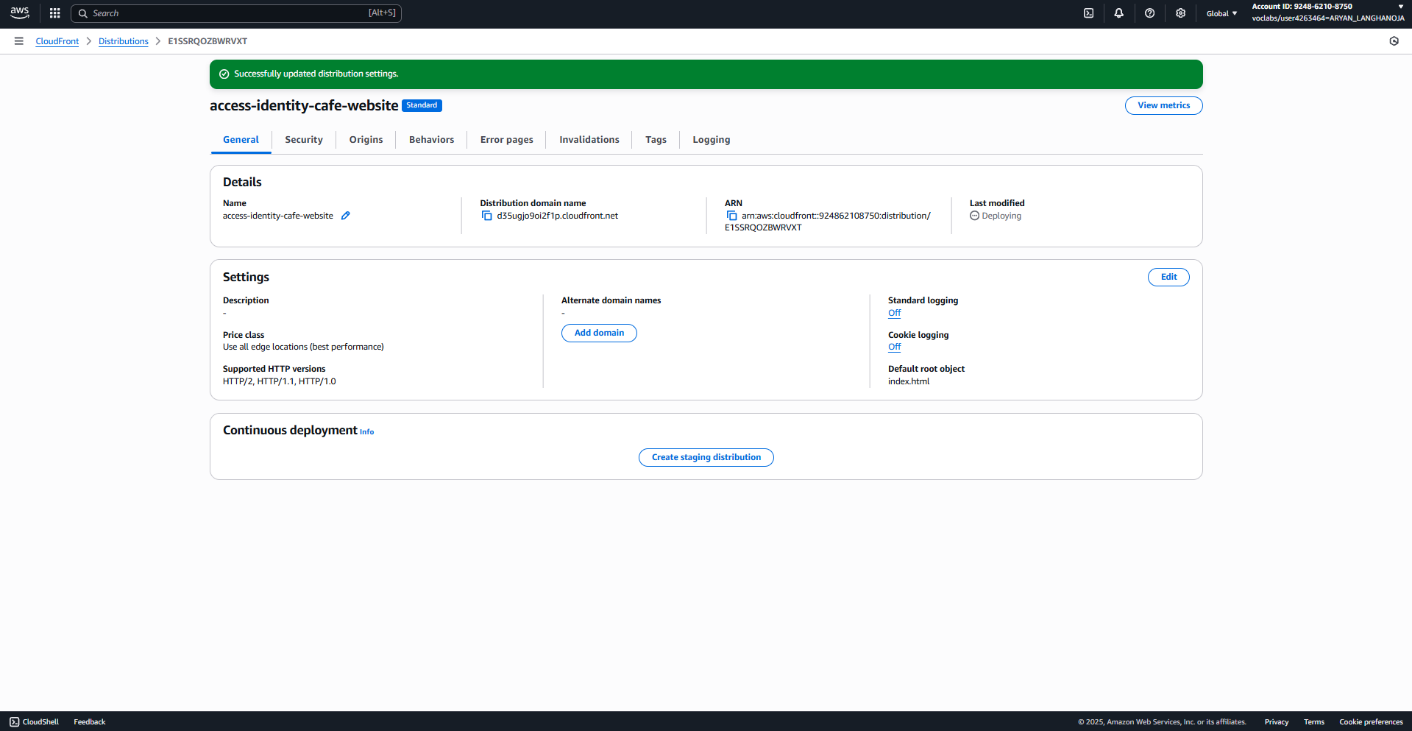
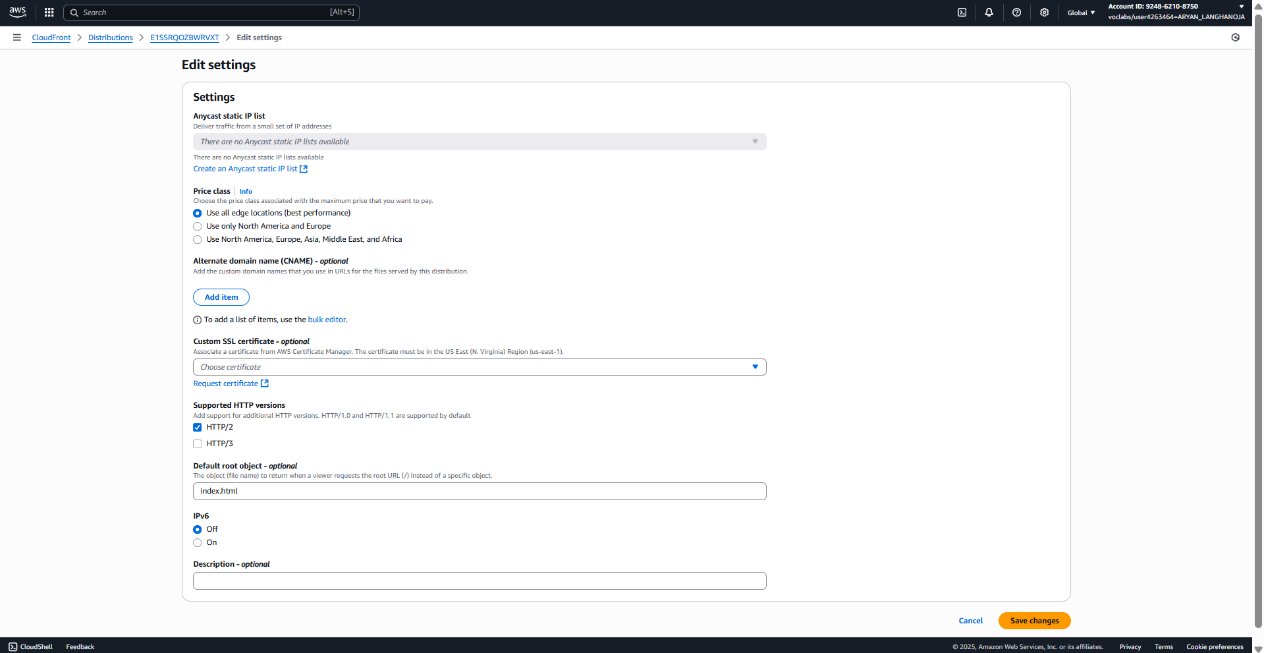
1. Configure the **Settings** section of the distribution.
   1. **Price class**: Keep the default **Use all edge locations** setting.
   2. **Alternative domain name (CNAME)**: Do not enter anything.

**Note**: You could specify a unique domain name here; however, this lab doesn't have one.

* 1. **Custom SSL certificate**: Choose **None** from dropdown.

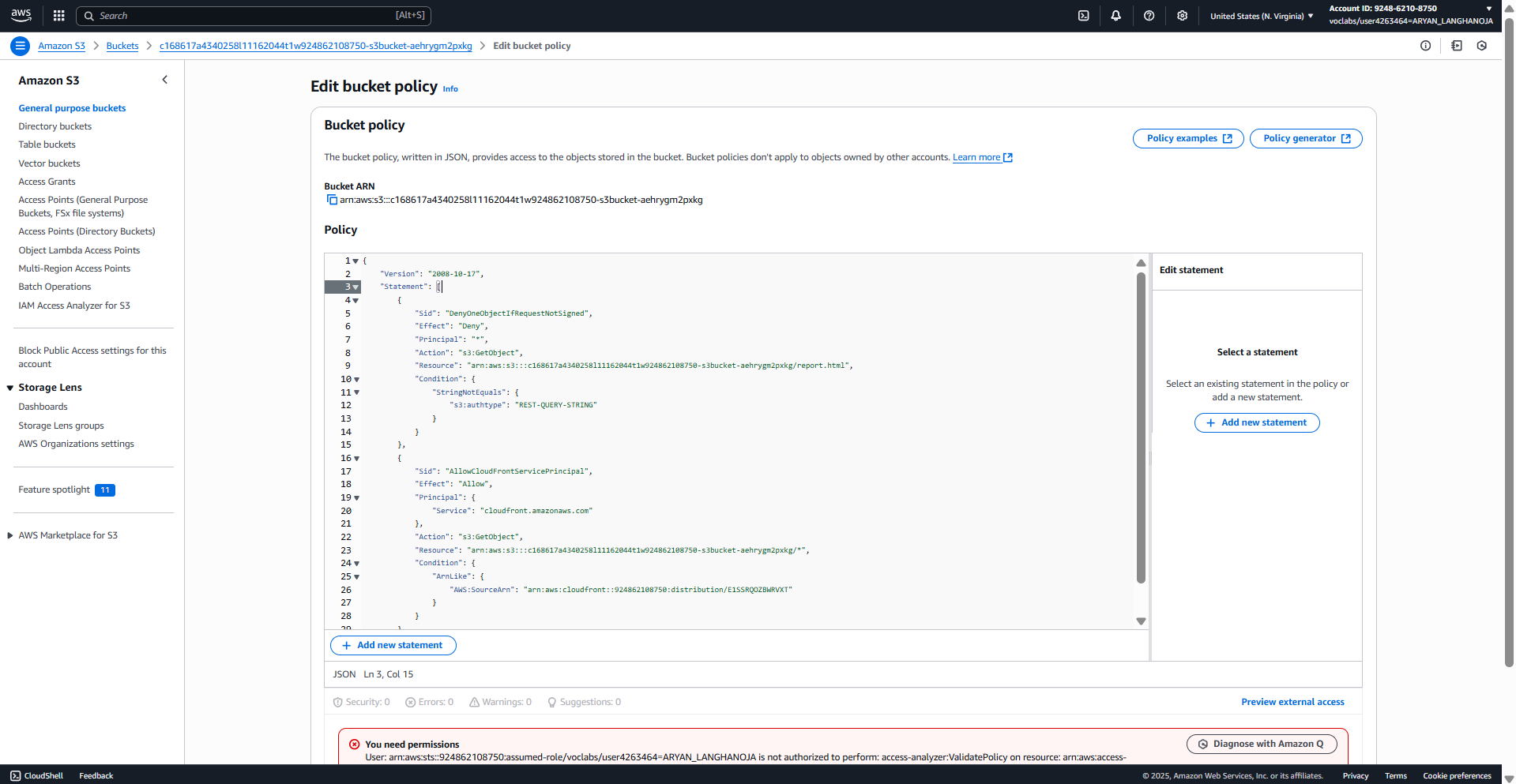
**Note**: If you owned a domain and had a TLS or SSL certificate from a certificate authority, you would upload it here. For this lab, you will use the default CloudFront certificate (\*.cloudfront.net), so you do not need to specify a custom certificate.

* 1. **Supported HTTP versions**: Select **HTTP/2**.
  2. **Default root object**: Enter index.html
  3. **IPv6**: Off

******Note**: This is the café website homepage hosted in the S3 bucket.

1. Update the bucket policy.

* In a new browser tab, navigate to the Amazon S3 console.
* Choose the link for the bucket that has *-s3bucket* in the name.
* Choose the **Permissions** tab.
* In the **Bucket policy** section, choose **Edit**.
* In the **Policy** code section, delete lines 4 through 17

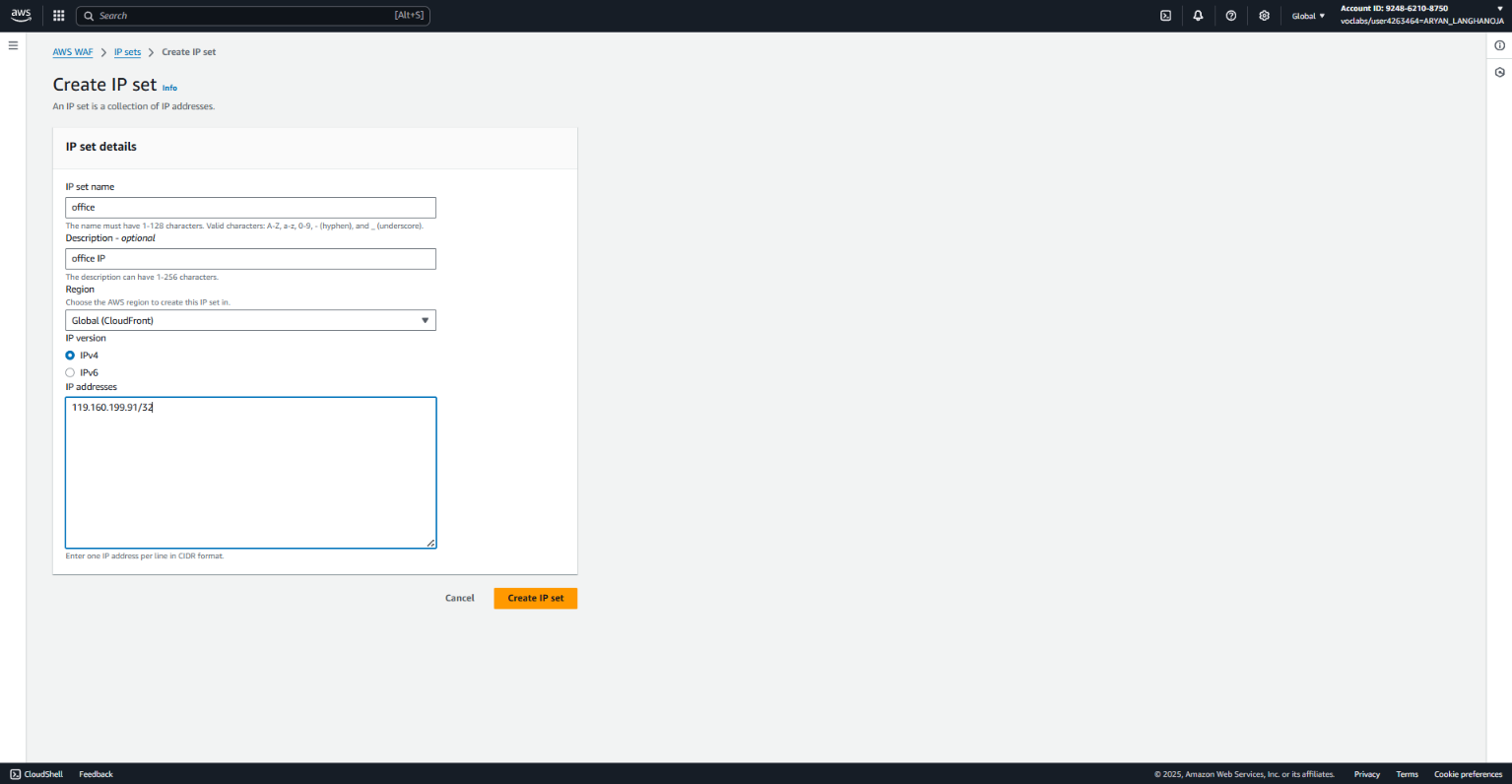


**Task 3: Securing network access to the distribution using AWS WAF**

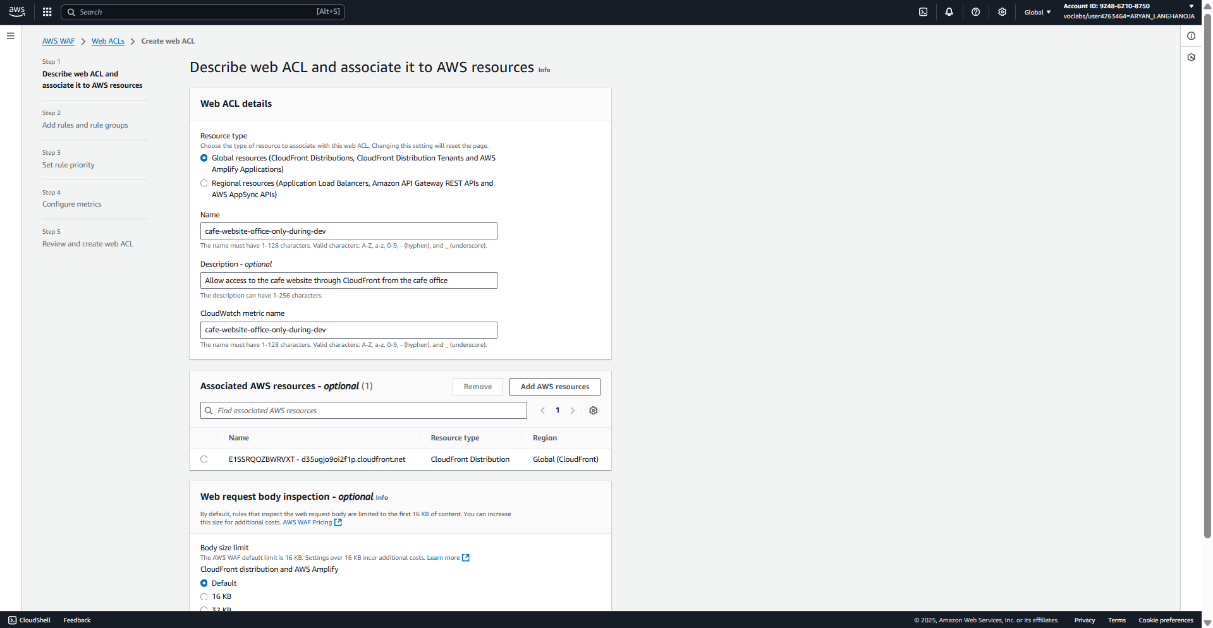
1. First, create an IP set for your IP address.
   1. In the AWS Management Console, search for waf and choose **WAF & Shield**.
   2. In the left navigation pane, choose **IP sets**.
   3. Choose **Create IP set** and configure:
      1. **IP set name**: Enter office
      2. **Description**: Enter office IP
      3. **Region**: Choose **Global (CloudFront)**.
      4. **IP addresses**: Enter <ip-address>/32 where <ip-address> is your public IPv4 address, as identified by whatismyipaddress.com.

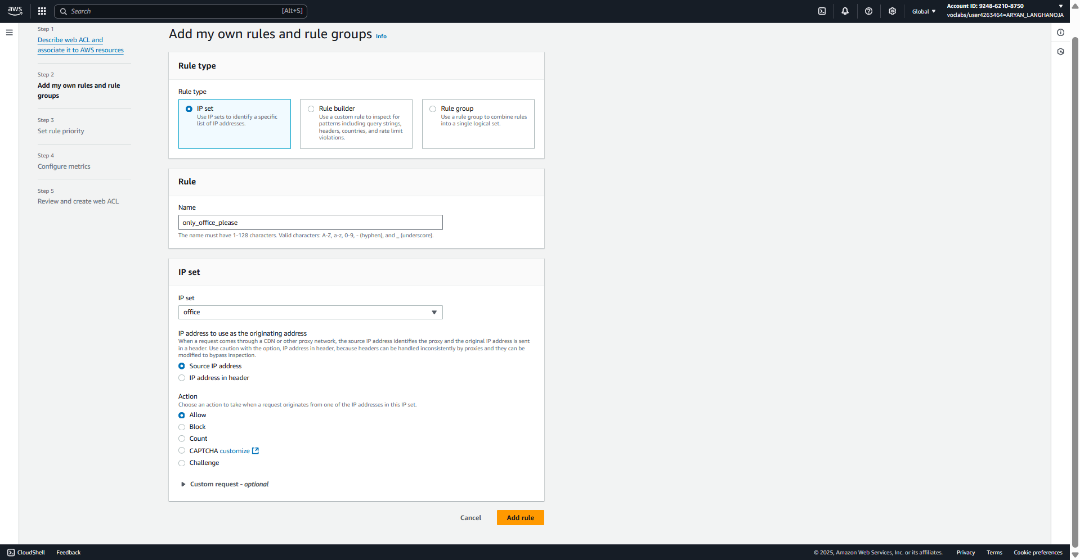
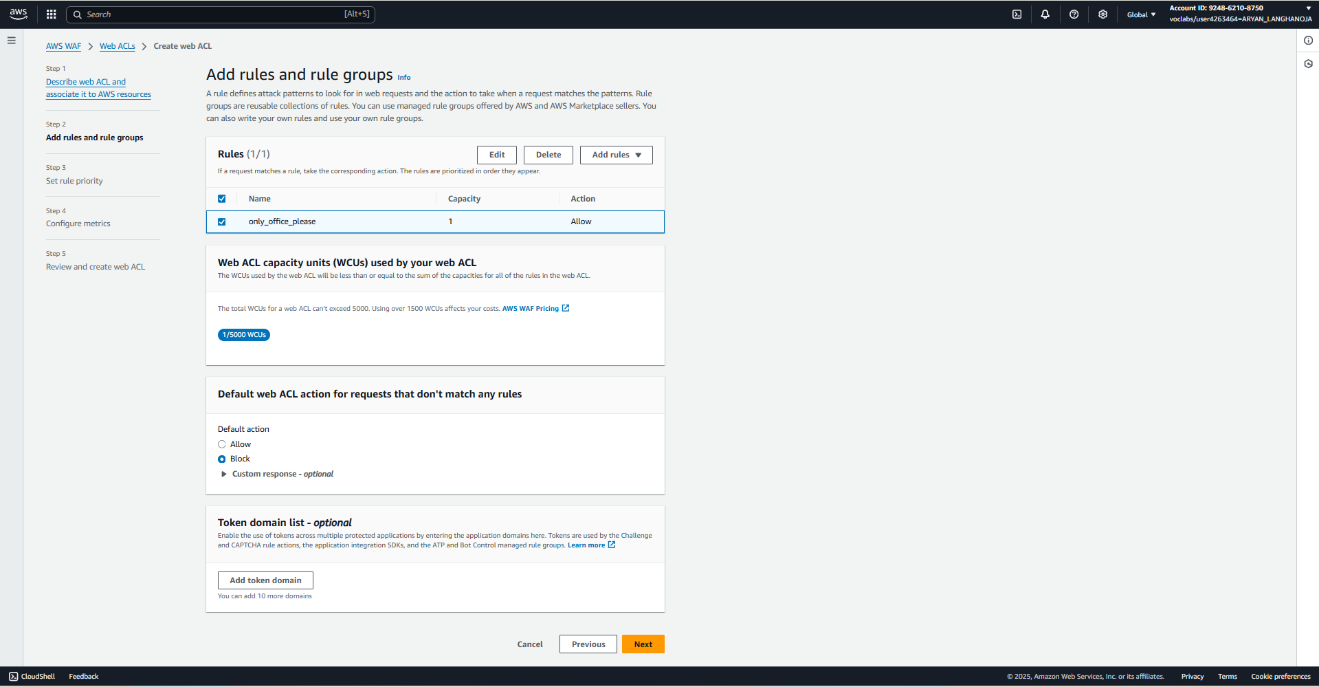
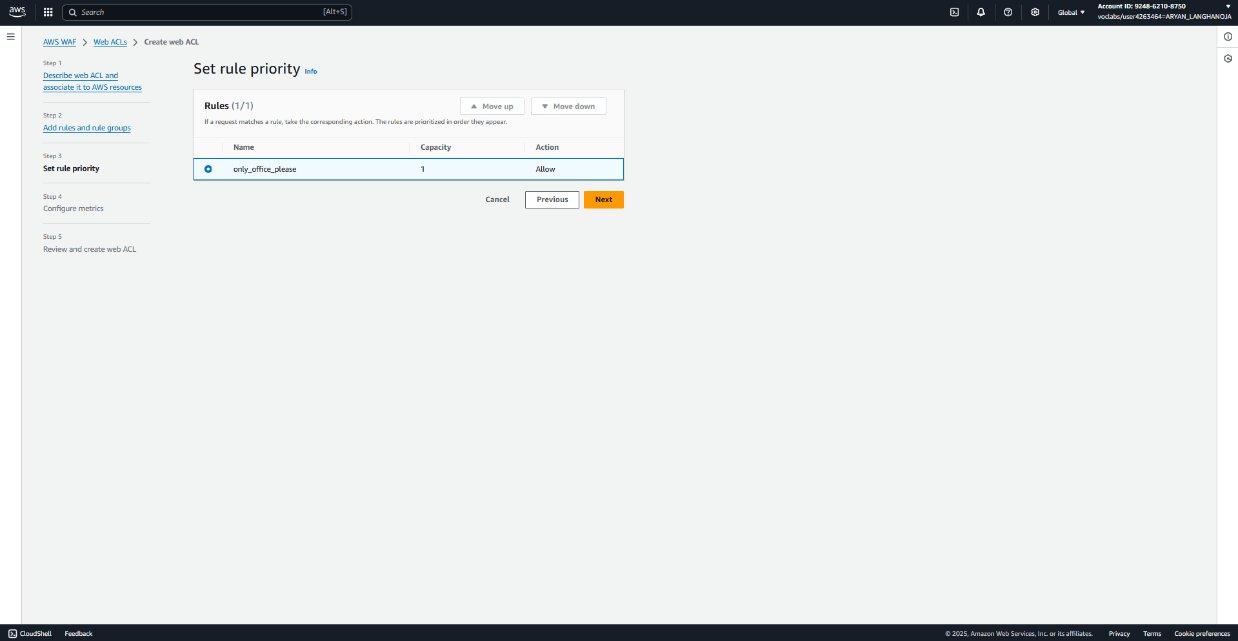
**Note**: Be sure to include the /32 at the end of the IP address.

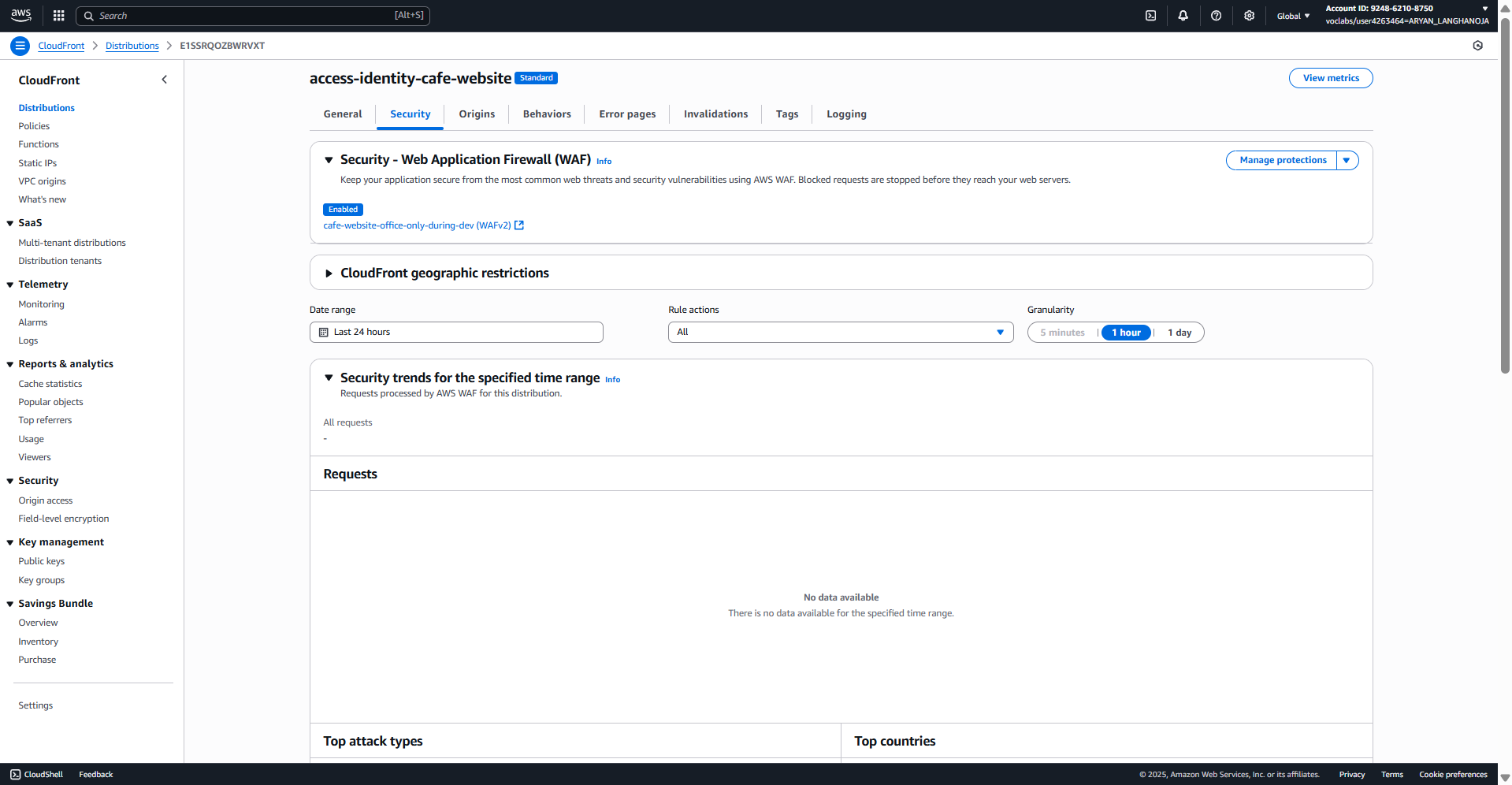
* + 1. Choose **Create IP set**.

****

1. Begin to create a web ACL.
   1. In the left navigation pane, choose **Web ACLs**.
   2. Choose **Create web ACL**.
   3. In the **Web ACL details** section, configure:
      1. **Resource type**: Choose **CloudFront distributions**.
      2. **Name**: Enter cafe-website-office-only-during-dev
      3. **Description**: Enter Allow access to the cafe website through CloudFront from the cafe office
      4. **CloudWatch metric name**: Enter cafe-website-office-only-during-dev
   4. In the **Associated AWS resources** section, configure:
      1. Choose **Add AWS resources**.
      2. Select the CloudFront distribution that you created.
      3. Choose **Add**.
      4. Select the CloudFront distribution again, and then choose **Next**.

****

1. Add a rule to the web ACL configuration to allow requests from the *office* IP set.
   1. In the **Rules** section, choose **Add rules**, **Add my own rules and rule groups**.
   2. **Rule type**: Choose **IP set**.
   3. **Name**: Enter only\_office\_please
   4. **IP set:** Choose the *office* IP set that you just created.
   5. **IP address to use as the originating address**: Keep the default **Source IP address** setting.
   6. **Action:** Choose **Allow**.
   7. Choose **Add rule**.
2. Update the new web ACL rule to block any requests that don't match the rule.
   1. In the **Rules** section, select the **only\_office\_please** rule.
   2. In the **Default web ACL action** section, for **Default action**, choose **Block**.
   3. Choose **Next**.
3. Set the rule priority, configure metrics, and create the web ACL.
   1. Choose the **only\_office\_please** rule.
   2. Choose **Next**.
   3. Keep all of the default metrics settings, and choose **Next** again.
   4. Review the settings, and at the bottom of the page, choose **Create web ACL**.
4. Confirm that the web ACL configuration has been applied to the CloudFront distribution.
   1. Return to the CloudFront console.
   2. In the left navigation pane, choose **Distributions**.
   3. **Note**: The **Last modified** column might display *Deploying* for the distribution. The deployment will complete within about 5 minutes; however, you can proceed to the next step without waiting.
   4. Choose the link for the distribution ID.
   5. In the **Security** section, notice that the distribution now has an **AWS WAF** value.

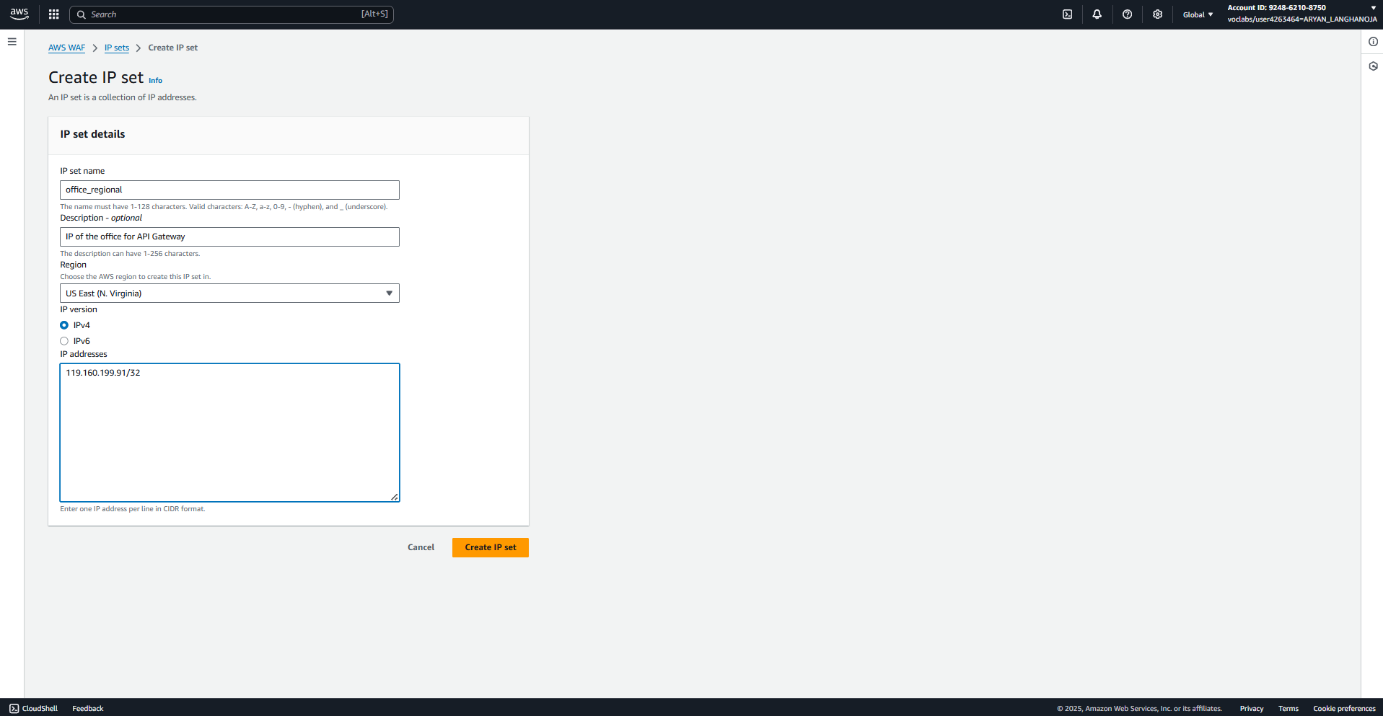


**Task 4: Securing a REST API endpoint using AWS WAF**

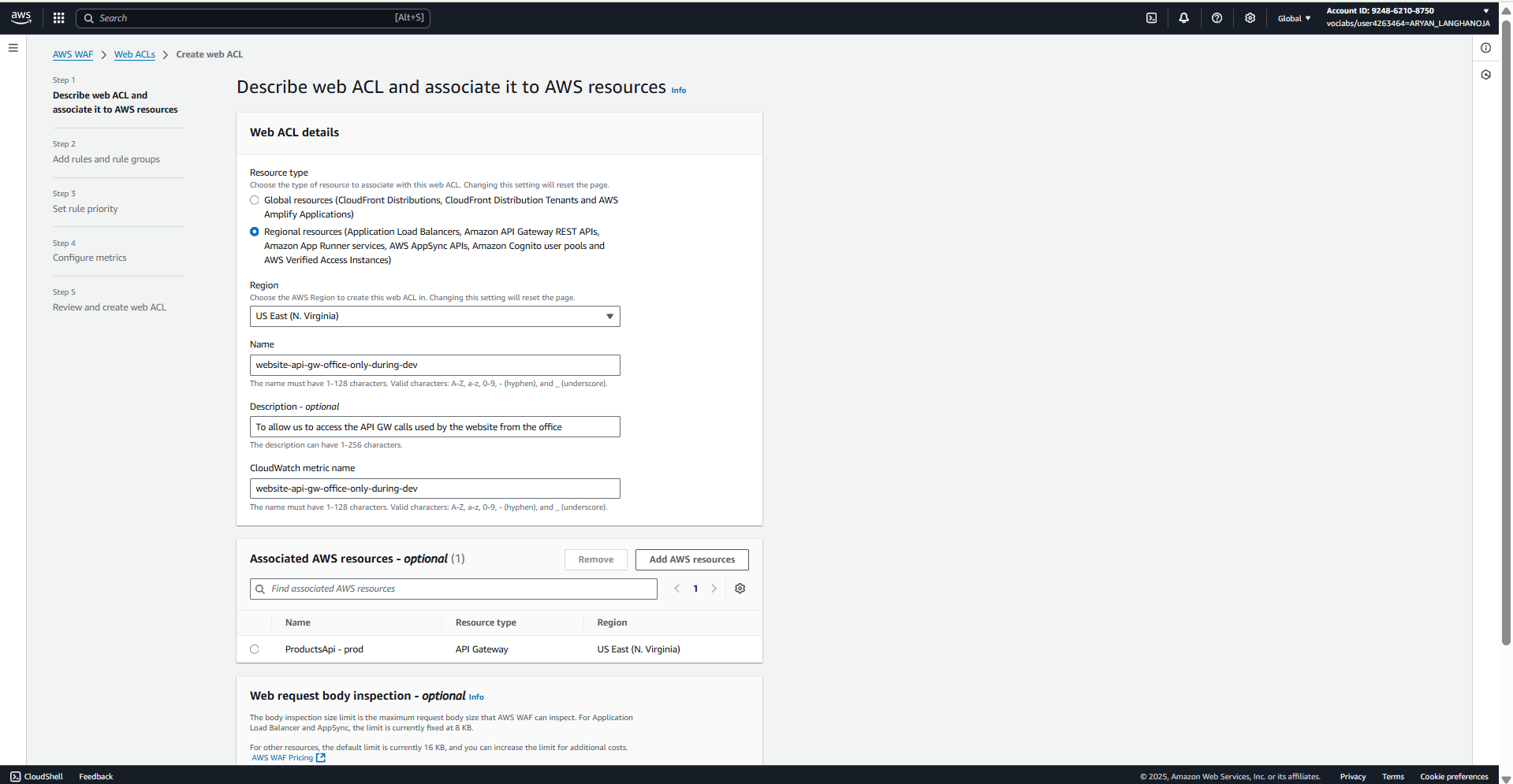
1. Create a *regional* AWS WAF IP set.
   1. Return to the WAF & Shield console.
   2. In the left navigation pane, choose **IP sets**.
   3. Choose **Create IP set** and configure:
      1. **IP set name**: Enter office\_regional
      2. **Description**: Enter IP of the office for API Gateway
      3. **Region**: Choose **US East (N. Virginia)**.
      4. **IP addresses**: Enter <ip-address>/32 where <ip-address> is your public IPv4 address, as identified by whatismyipaddress.com.

**Note**: Be sure to include the /32 at the end of the IP address.

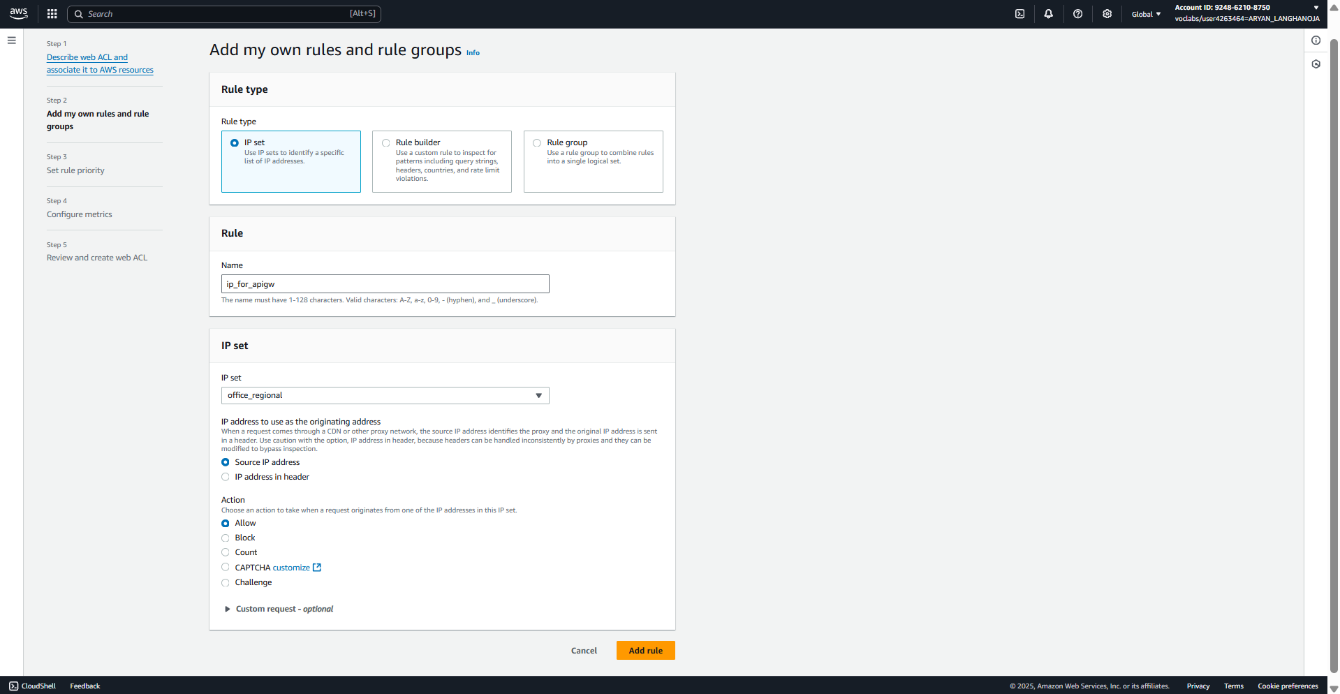
* 1. Choose **Create IP set**.

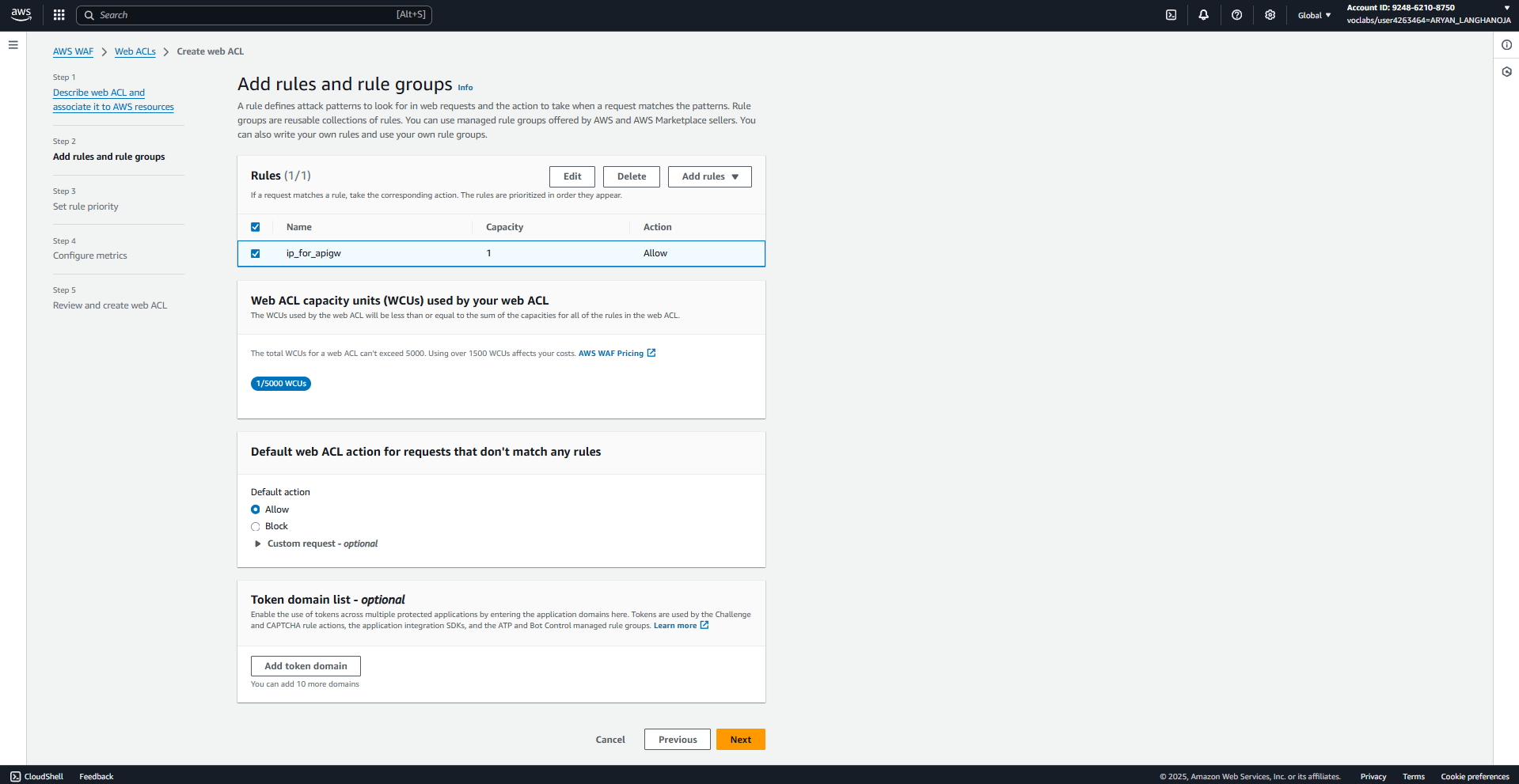
****

1. Begin to create a *regional* web ACL.
   1. In the left navigation pane, choose **Web ACLs**.
   2. Choose **Create web ACL**.
   3. In the **Web ACL details** section, configure:
      1. **Resource type**: Choose **Regional resources**.
      2. **Region**: Choose **US East (N. Virginia)**.
      3. **Name**: Enter website-api-gw-office-only-during-dev
      4. **Description**: Enter To allow us to access the API GW calls used by the website from the office
      5. **CloudWatch metric name**: Enter website-api-gw-office-only-during-dev
   4. In the **Associated AWS resources** section, configure:
      1. Choose **Add AWS resources**.
      2. For Resource type, select **Amazon API Gateway**.
      3. Select the **ProductsApi - prod** API Gateway resource.
      4. Choose **Add**.
      5. Select the **ProductsApi - prod** resource again, and then choose **Next**.

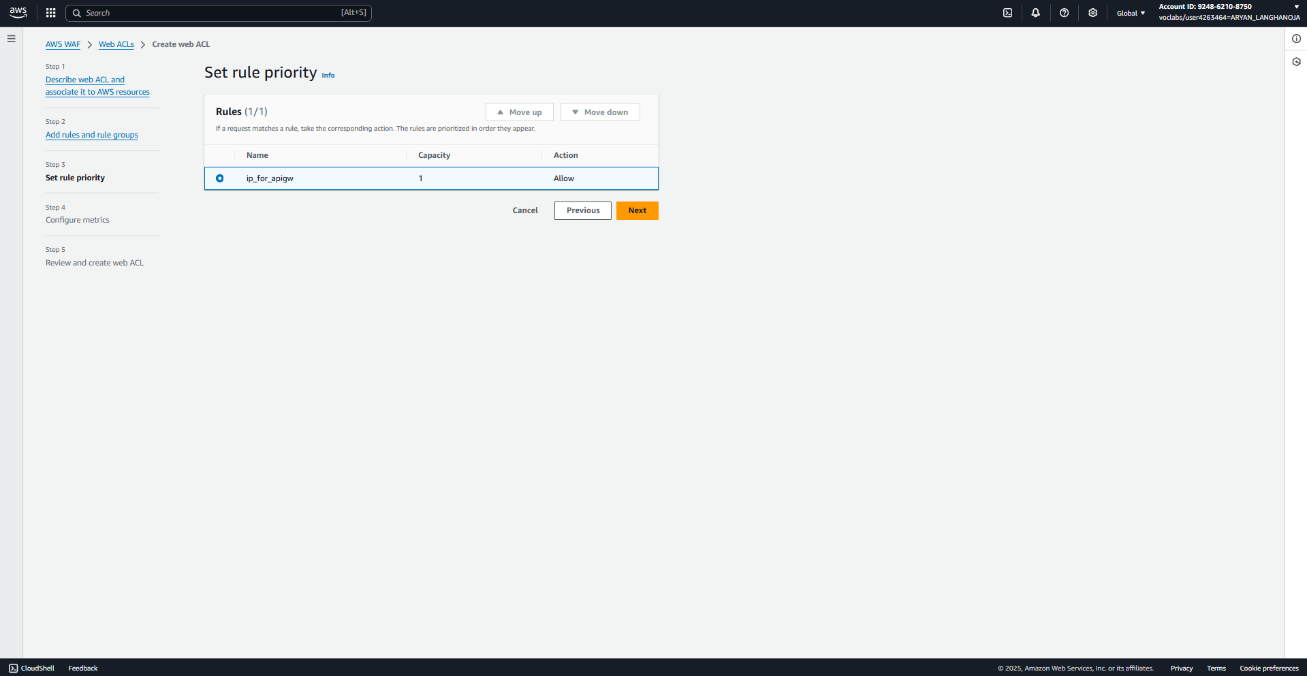


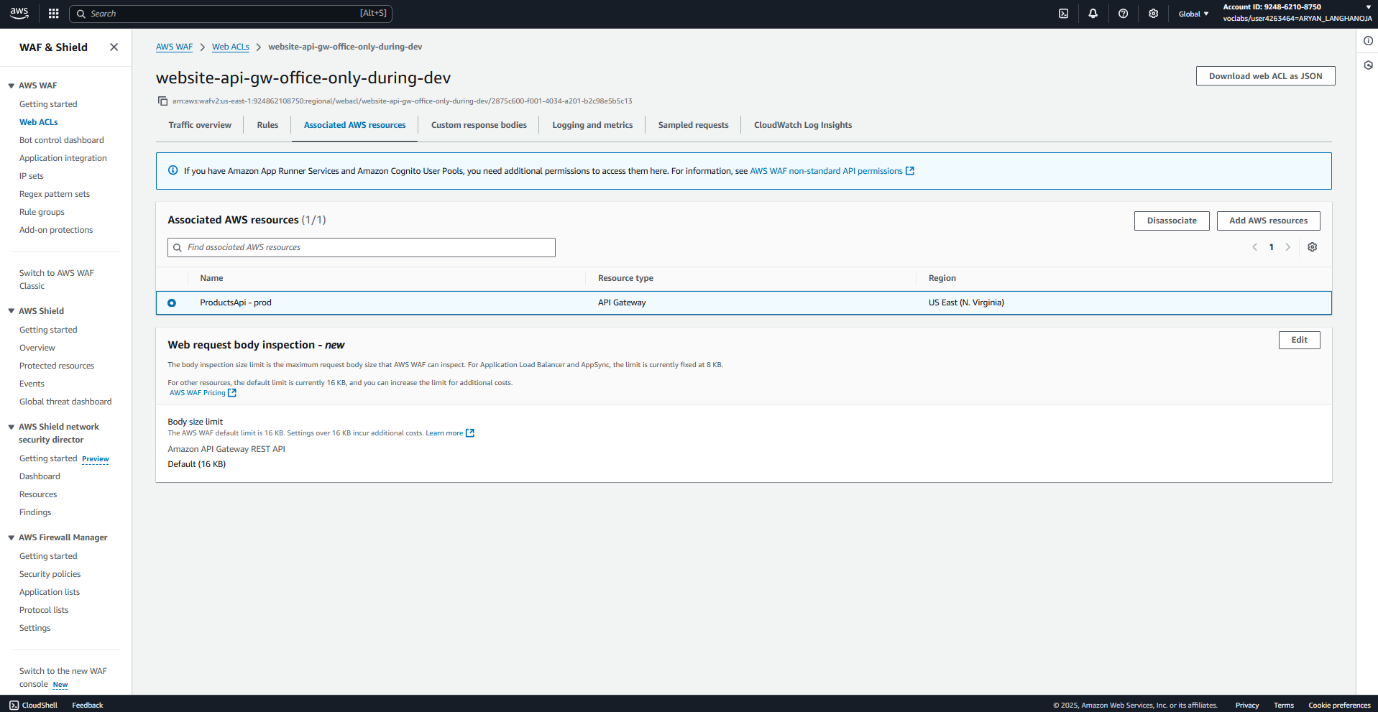
1. Add a rule to the web ACL configuration to allow requests from API Gateway.
   1. In the **Rules** section, choose **Add rules**, **Add my own rules and rule groups**.
   2. **Rule type**: Choose **IP set**.
   3. **Name**: Enter ip\_for\_apigw
   4. **IP set**: Choose the *office\_regional* IP set that you just created.
   5. **IP address to use as the originating address**: Keep the default **Source IP address** setting.
   6. **Action**: Choose **Allow**.
   7. Choose **Add rule**.

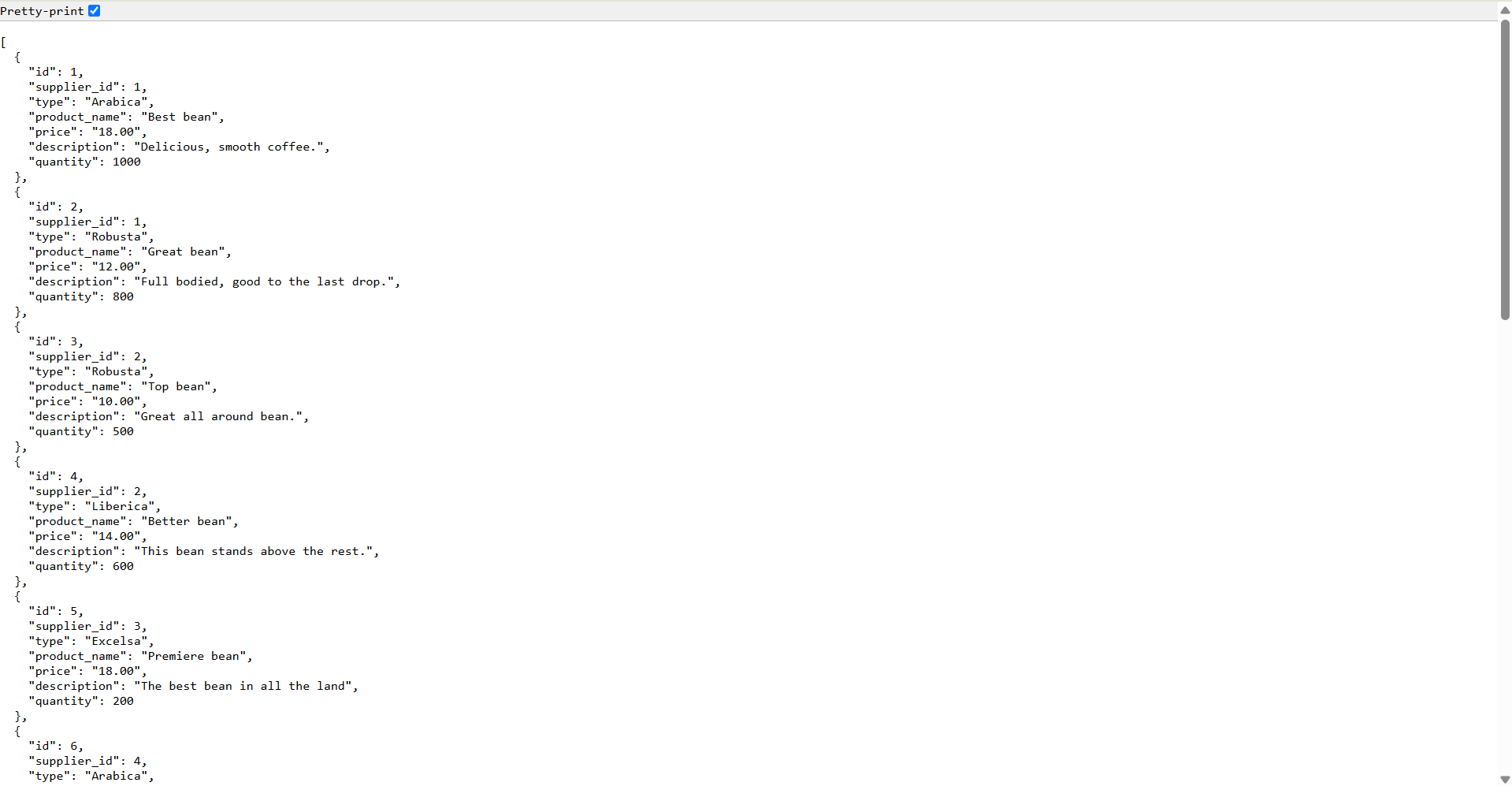


1. Update the new web ACL rule to block any requests that don't match the rule.
   1. In the **Rules** section, select the **ip\_for\_apigw** rule.
   2. In the **Default web ACL action** section, for **Default action**, choose **Block**.
   3. Choose **Next**.

1. Set the rule priority, configure metrics, and create the web ACL.
   1. Choose the **ip\_for\_apigw** rule.
   2. Choose **Next**.
   3. Keep all of the default metrics settings, and choose **Next** again.
   4. Review the settings, and at the bottom of the page, choose **Create web ACL**.



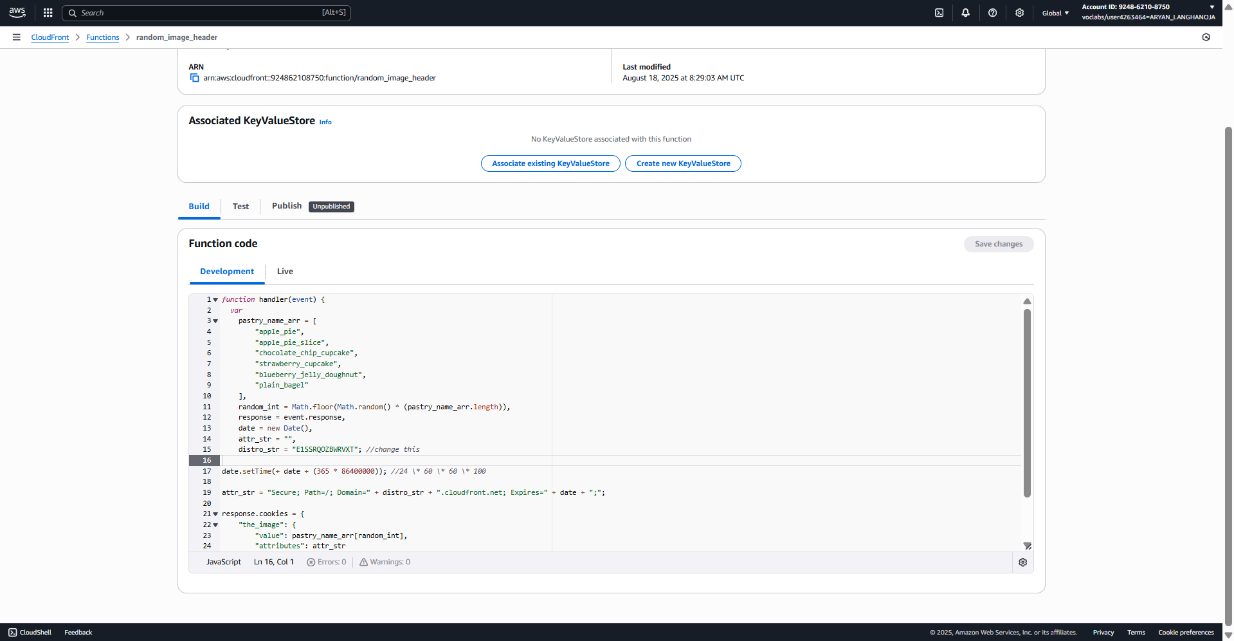
1. When the web ACL creation process is complete, check the resources associated with the ACL.
   1. Choose the link for the *website-api-gw-office-only-during-dev* ACL, which you just created.
   2. Choose the **Associated AWS resources** tab.
   3. Confirm that the *ProductsApi - prod* resource is listed. If it isn't, add it:
      1. Choose **Add AWS resources**.
      2. Choose the *ProductsApi - prod* resource.
      3. Choose **Add**.
2. Test the new ACL from your computer.

* In a new browser tab, go to the API Gateway console.
* Choose the link for the **ProductsApi**.
* In the left navigation pane, choose **Stages**.
* In the **Stages** navigation pane, expand the **prod** stage.
* Under **/bean\_products**, choose **GET**.
* Copy the **Invoke URL**, which has the format *https://.execute-api.us-east-1.amazonaws.com/prod/bean\_products*, and load the URL in a new browser tab.
* A JSON-formatted document with product information displays. This is the expected behavior.

**Task 5: Configuring a CloudFront function on the website**

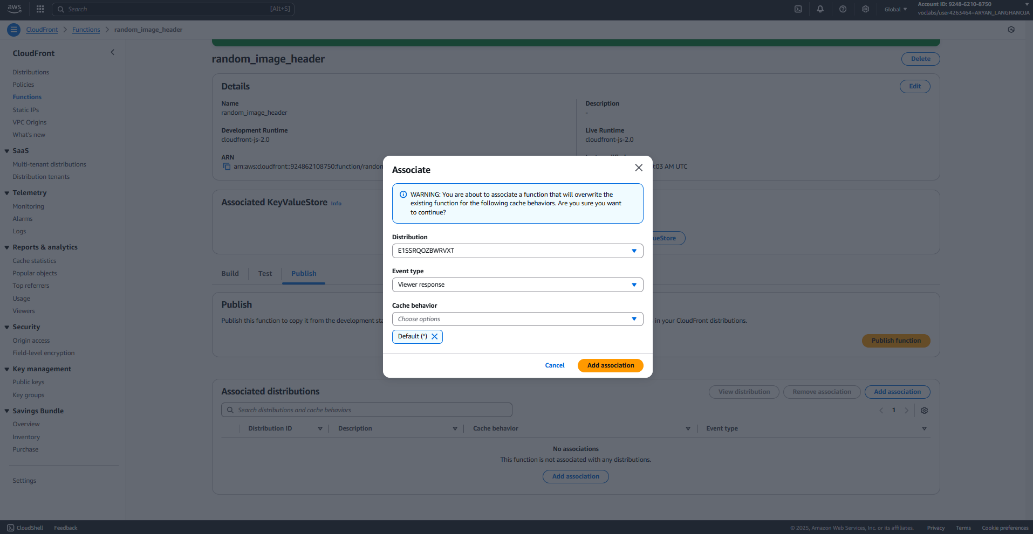
1. Create a CloudFront function.

* Navigate to the CloudFront console.
* In the left navigation pane, choose **Functions**.
* Choose **Create function**.
* For **Name**, enter random\_image\_header
* Choose **Create function**.

****

1. Publish the CloudFront function, and associate it with the CloudFront distribution.

* Choose the **Publish** tab.
* Choose **Publish function**.
* A message at the top of the page indicates that the *random\_image\_header* function was successfully published.
* In the **Associated distributions** section, choose **Add association** and configure:
  + **Distribution**: Choose the distribution.
  + **Event type**: Choose **Viewer Response**.
  + **Cache behavior**: Select **Default (\*)**.
  + Choose **Add association**.
* In the left navigation pane, choose **Distributions**.
* Notice that the distribution is being deployed again.



**Task 6: Adjusting the cache duration**

1. Edit the Cache-Control header set on each object in the S3 bucket.

* Go to the VS Code IDE bash terminal and run the following command:

**BUCKET\_NAME=$(aws s3api list-buckets --query "Buckets[?contains(Name, 's3bucket')].Name" --output text)**

**echo $BUCKET\_NAME**

**aws s3 ls s3://$BUCKET\_NAME/ | awk '{print $4}' | xargs -I {} aws s3api copy-object \**

**--bucket $BUCKET\_NAME \**

**--copy-source $BUCKET\_NAME/{} \**

**--key {} \**

**--cache-control "max-age=180" \**

 **--metadata-directive REPLACE**

**Conclusion:-**

* I connected to the AWS IDE and set up the lab environment.
* I downloaded the required files and upgraded the AWS CLI.
* I verified the S3 bucket and opened the café website.
* I created a CloudFront distribution for the website.
* I enabled HTTPS access using CloudFront.
* I updated the S3 bucket policy for secure access.
* I created an IP set in AWS WAF for my system IP.
* I created a Web ACL to allow only my IP to access the site.
* I secured the API Gateway endpoint using AWS WAF.
* I tested the API Gateway URL and verified it was working.
* I created and deployed a CloudFront function.
* I updated cache settings on S3 objects for better performance.