Autoscaling Configuration for Backend and Frontend Infrastructure on AWS

Backend Autoscaling

Step 1: Setting up the Backend AMI

1. **Launch a new EC2 instance**:
   - Install and configure the backend application (Node.js, Django, etc.) and any dependencies.
   - Validate the setup to ensure the backend functions correctly on HTTP port 8000.
2. **Create an AMI from the EC2 instance**:
   - Go to **EC2 Dashboard** > **Instances**.
   - Select the instance and click **Actions** > **Create Image**.
   - Name the AMI appropriately (e.g., Backend_AMI).
   - This AMI will serve as the template for launching instances in the Auto Scaling group.

Step 2: Create Launch Template for Backend Instances

1. **Go to Launch Templates** in the **EC2 Dashboard**.
2. **Create Launch Template**:
   - Provide a name (e.g., Backend_Launch_Template).
   - Under **AMI ID**, select the Backend_AMI created in Step 1.
   - Choose an instance type appropriate for the backend.
   - Configure the template to use the backend security group (created in Step 5).
   - Optionally, configure other details like network settings and IAM roles.

Step 3: Setting up the Target Group for Backend Instances

1. **Create a Target Group**:
   - Go to **Target Groups** in the **EC2 Dashboard** > **Create Target Group**.
   - Choose **Instances** as the target type.

    - Set **Protocol** to **HTTP** and **Port** to **8000**.
    - Select the appropriate **VPC** and configure **Health Checks** to monitor the health of instances on HTTP port 8000.
    - Leave the **Register Targets** section blank, as instances will be added dynamically by the Auto Scaling Group.

Step 4: Configuring the Internal Application Load Balancer for Backend

1. **Create an Internal ALB**:
    - Go to **Load Balancers** > **Create Load Balancer**.
    - Select **Application Load Balancer** and set it to **Internal** for VPC-only access.
    - Configure listeners to listen on **Port 80**.
2. **Attach the Backend Target Group to the ALB**:
    - Under **Listeners and Routing**, add a rule to forward traffic from **Port 80** to the **Backend Target Group** created in Step 3.
    - Assign the ALB to **private subnets** across multiple availability zones for high availability.

Step 5: Configure Backend Security Group

1. **Create a Security Group for the Backend ALB**:
    - Go to **Security Groups** in **VPC Dashboard**.
    - Create a group (e.g., backend-alb-sg) that allows **HTTP (port 80)** traffic from the frontend ALB security group.
2. **Configure Security Group for Backend EC2 Instances**:
    - Create another Security Group for backend instances (e.g., backend-instances-sg).
    - Allow **HTTP (port 8000)** traffic from the backend ALB security group.
    - Restrict outbound access as necessary to maintain security.

Step 6: Create an Autoscaling Group for the Backend

- Go to **EC2 Console** > **Auto Scaling Groups** > **Create Auto Scaling Group**.
- Use the launch template created in Step 2.
- Attach the internal load balancer and backend target group to ensure load distribution.
- Set desired capacity, scaling policies, and health checks as needed.
- Complete the configuration to launch and manage backend instances.

Frontend Autoscaling

Step 1: Setting up the Frontend AMI

1. **Launch a new EC2 instance**:
   - Set up the frontend application (Apache, NGINX, etc.) and any required configurations.
   - Validate that the frontend application works correctly on HTTP port 80.
2. **Create an AMI from the EC2 instance**:
   - Go to **EC2 Dashboard** > **Instances**.
   - Select the instance and click **Actions** > **Create Image**.
   - Name this AMI (e.g., Frontend_AMI), which will serve as the template for the frontend Auto Scaling Group.

Step 2: Create Launch Template for Frontend Instances

1. **Go to Launch Templates** in the **EC2 Dashboard**.
2. **Create Launch Template**:
   - Provide a name (e.g., Frontend_Launch_Template).
   - Select the Frontend_AMI created in Step 1.
   - Choose an instance type suitable for the frontend.
   - Configure the template to use the frontend security group (configured in Step 5).

Step 3: Setting up the Target Group for Frontend Instances

1. **Create a Target Group for Frontend**:
   - Go to **Target Groups** in **EC2 Dashboard** > **Create Target Group**.
   - Select **Instances** as the target type and set **Protocol** to **HTTP** with **Port 80**.
   - Configure **Health Checks** to monitor HTTP traffic on Port 80.
   - Leave **Register Targets** empty to let the Auto Scaling Group handle instance registration.

Step 4: Configuring the Public Application Load Balancer for Frontend

1. **Create a Public ALB**:
   - Go to **Load Balancers** > **Create Load Balancer**.
   - Choose **Application Load Balancer** and select **Internet-facing** to allow external access.
   - Configure listeners to listen on **Port 80**.
2. **Attach the Frontend Target Group to the Public ALB**:
   - Under **Listeners and Routing**, forward traffic from **Port 80** to the **Frontend Target Group** created in Step 3.
   - Assign the ALB to **public subnets** in multiple availability zones to ensure high availability.

Step 5: Configure Frontend Security Group

1. **Create a Security Group for the Public ALB**:
   - In **Security Groups** under **VPC Dashboard**, create a group (e.g., frontend-alb-sg).
   - Allow **HTTP (port 80)** from **0.0.0.0/0** to enable public internet access.
2. **Configure Security Group for Frontend EC2 Instances**:
   - Create another Security Group (e.g., frontend-instances-sg) for frontend instances.
   - Allow **HTTP (port 80)** traffic from the frontend ALB security group.
   - Limit outbound access as needed for security.

Step 6: Create an Autoscaling Group for the Frontend

- Go to **EC2 Console** > **Auto Scaling Groups** > **Create Auto Scaling Group**.
- Select the launch template created in Step 2 for frontend scaling.
- Attach the public load balancer and frontend target group.
- Set desired capacity, scaling policies, and health checks.
- Finalize settings to launch and manage frontend instances in the autoscaling group.

Verification and Testing

Step 1: Simulate High Load on an Instance

1. **SSH into an EC2 Instance in the Autoscaling Group:**
   - Use SSH to connect to one of the instances created by the autoscaling group.

     ```
     ssh -i /path/to/key.pem ec2-user@<instance_public_ip>
     ```

2. **Increase CPU Load with stress Command:**
   - Install stress if it's not already installed:

     ```
     sudo apt-get install -y stress  # For Ubuntu
     ```

   - Run the following command to increase CPU load:

     ```
     stress --cpu 6 -t 600
     ```

   - This command will simulate high CPU usage across 6 cores for 10 minutes.
3. **Monitor CPU Utilization in Another Terminal:**
   - In a separate terminal, SSH into the same instance and run:

     ```
     top
     ```

   - Observe CPU utilization, and confirm that it crosses the 50% threshold, which should trigger the autoscaling policy.

Step 2: Verify Autoscaling Triggers New Instance Launches

1. **Check AWS Console for New Instances:**
   - Go to the **EC2 Console** > **Auto Scaling Groups** > **Activity** to monitor the scaling activities.
   - You should see new instances being launched by the autoscaling group due to high CPU usage on the original instance.

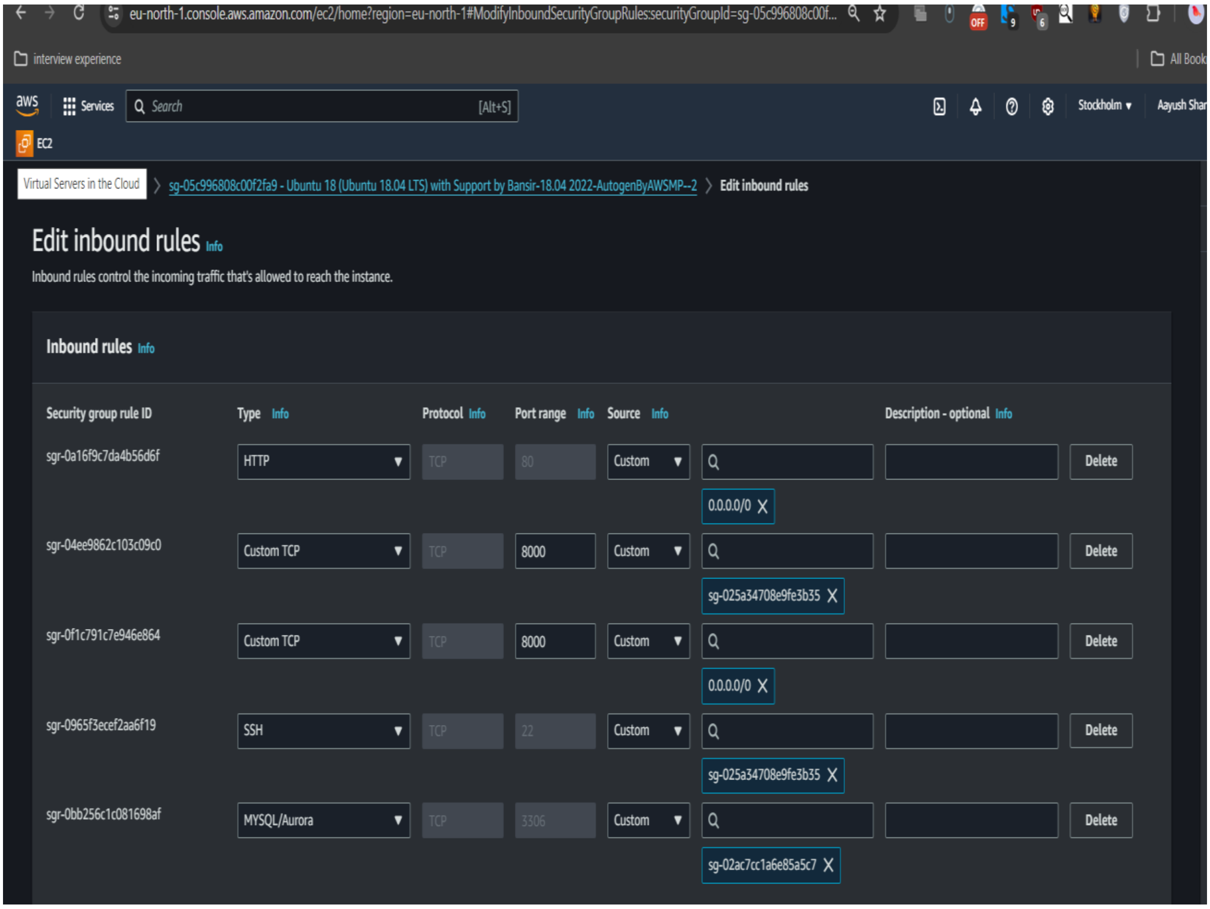
Step 3: Manually Test Autoscaling by Stopping the Original Instance

1. **Stop the Original Instance:**
   - In the **EC2 Console**, select the original instance, and stop it (do not terminate).
   - This will simulate an instance failure, causing the autoscaling group to replace the stopped instance.
2. **Check Application Accessibility:**
   - Access the application through the load balancer endpoint to ensure that the new instances created by autoscaling are handling the load.

- This confirms that the autoscaling configuration is correctly set to handle instances going down or high CPU loads.

Security group configuration

1.) backend instance (sg-05c996808c00f2fa9)



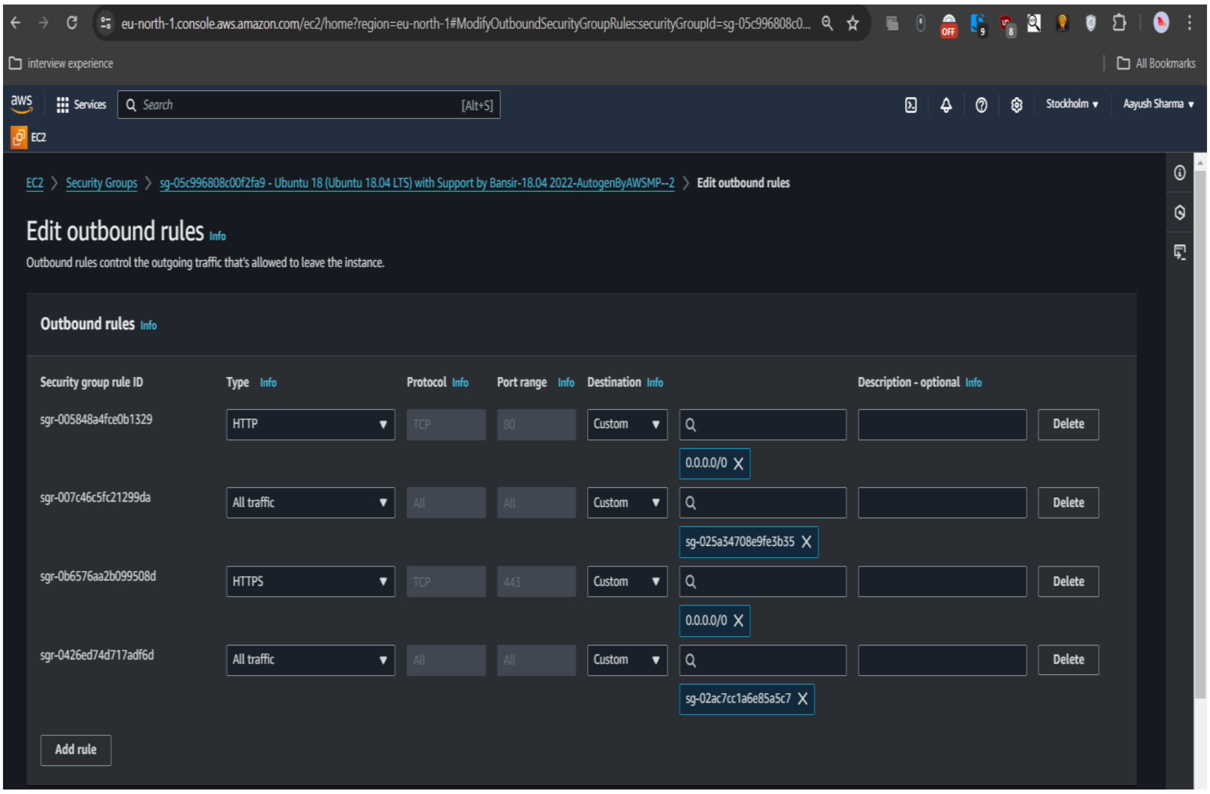
eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-05c996808c00f...
interview experience
All Book
aws
Services
Search
[Alt+S]
Stockholm ▾
Aayush Shar
EC2
Virtual Servers in the Cloud
sg-05c996808c00f2fa9 - Ubuntu 18 (Ubuntu 18.04 LTS) with Support by Bansir-18.04 2022-AutogenByAWSMP--2
Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0a16f9c7da4b56d6f	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-04ee9862c103c09c0	Custom TCP	TCP	8000	Custom	sg-025a34708e9fe3b35		Delete
sgr-0f1c791c7e946e864	Custom TCP	TCP	8000	Custom	0.0.0.0/0		Delete
sgr-0965f3ecef2aa6f19	SSH	TCP	22	Custom	sg-025a34708e9fe3b35		Delete
sgr-0bb256c1c081698af	MYSQL/Aurora	TCP	3306	Custom	sg-02ac7cc1a6e85a5c7		Delete



eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#ModifyOutboundSecurityGroupRules:securityGroupId=sg-05c996808c0...
interview experience
All Bookmarks
aws
Services
Search
[Alt+S]
Stockholm ▾
Aayush Sharma ▾
EC2

EC2 > Security Groups > sg-05c996808c00f2fa9 - Ubuntu 18 (Ubuntu 18.04 LTS) with Support by Bansir-18.04 2022-AutogenByAWSMP--2 > Edit outbound rules

Edit outbound rules Info

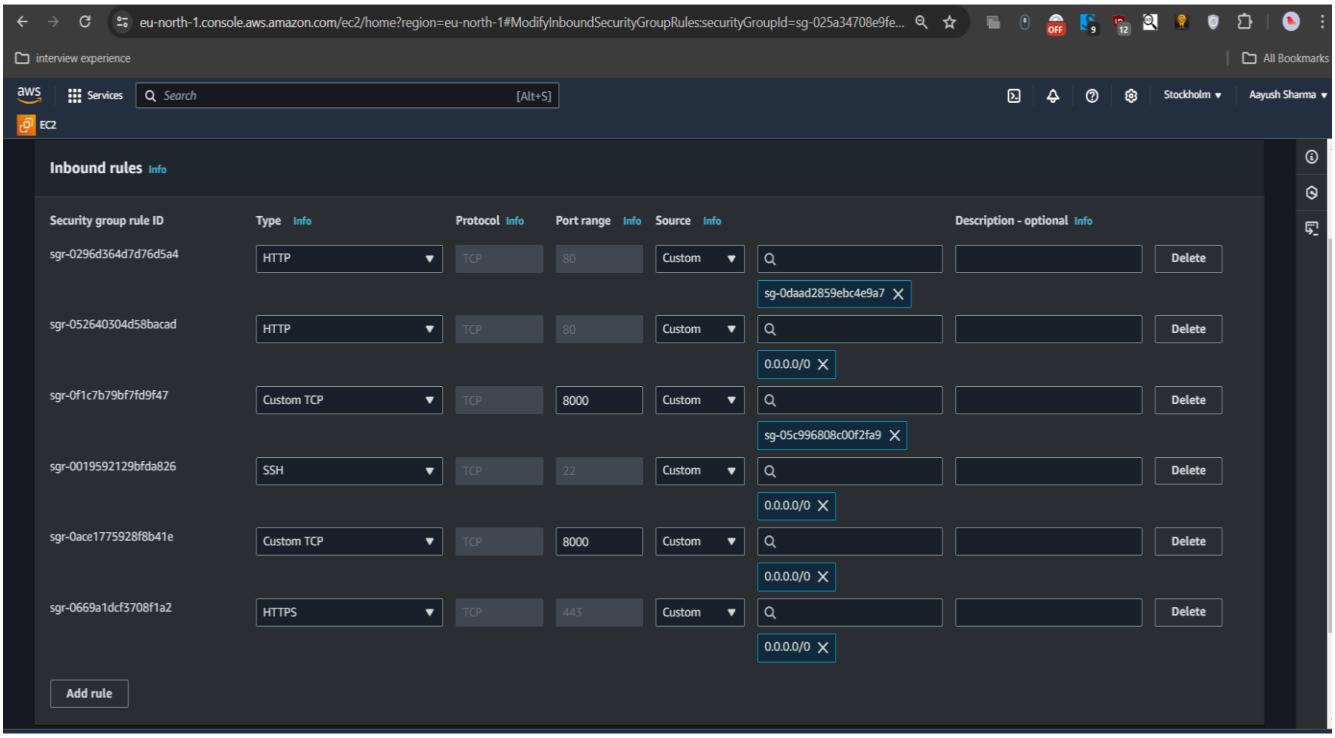
Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Destination Info		Description - optional Info	
sgr-005848a4fce0b1329	HTTP ▾	TCP	80	Custom ▾	0.0.0.0/0		Delete
sgr-007c46c5fc21299da	All traffic ▾	All	All	Custom ▾	sg-025a34708e9fe3b35		Delete
sgr-0b6576aa2b099508d	HTTPS ▾	TCP	443	Custom ▾	0.0.0.0/0		Delete
sgr-0426ed74d717adf6d	All traffic ▾	All	All	Custom ▾	sg-02ac7cc1a6e85a5c7		Delete

Add rule

2.) Frontend instance (sg-025a34708e9fe3b35)

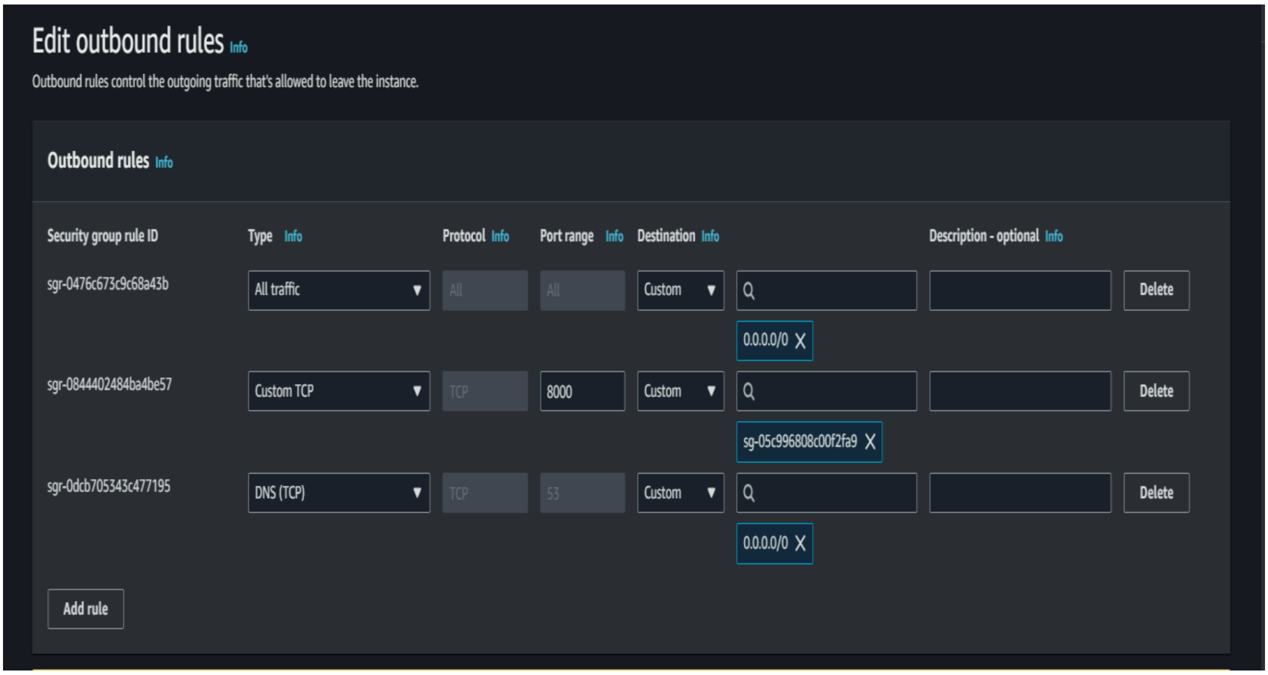


eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-025a34708e9fe...
interview experience
All Bookmarks
aws
Services
Search
[Alt+S]
Stockholm ▾
Aayush Sharma ▾
EC2

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0296d364d7d76d5a4	HTTP ▾	TCP	80	Custom ▾	sg-0daad2859ebc4e9a7		Delete
sgr-052640304d58bacad	HTTP ▾	TCP	80	Custom ▾	0.0.0.0/0		Delete
sgr-0f1c7b79bf7fd9f47	Custom TCP ▾	TCP	8000	Custom ▾	sg-05c996808c00f2fa9		Delete
sgr-0019592129bfda826	SSH ▾	TCP	22	Custom ▾	0.0.0.0/0		Delete
sgr-0ace1775928f8b41e	Custom TCP ▾	TCP	8000	Custom ▾	0.0.0.0/0		Delete
sgr-0669a1dcf3708f1a2	HTTPS ▾	TCP	443	Custom ▾	0.0.0.0/0		Delete

Add rule



Edit outbound rules Info

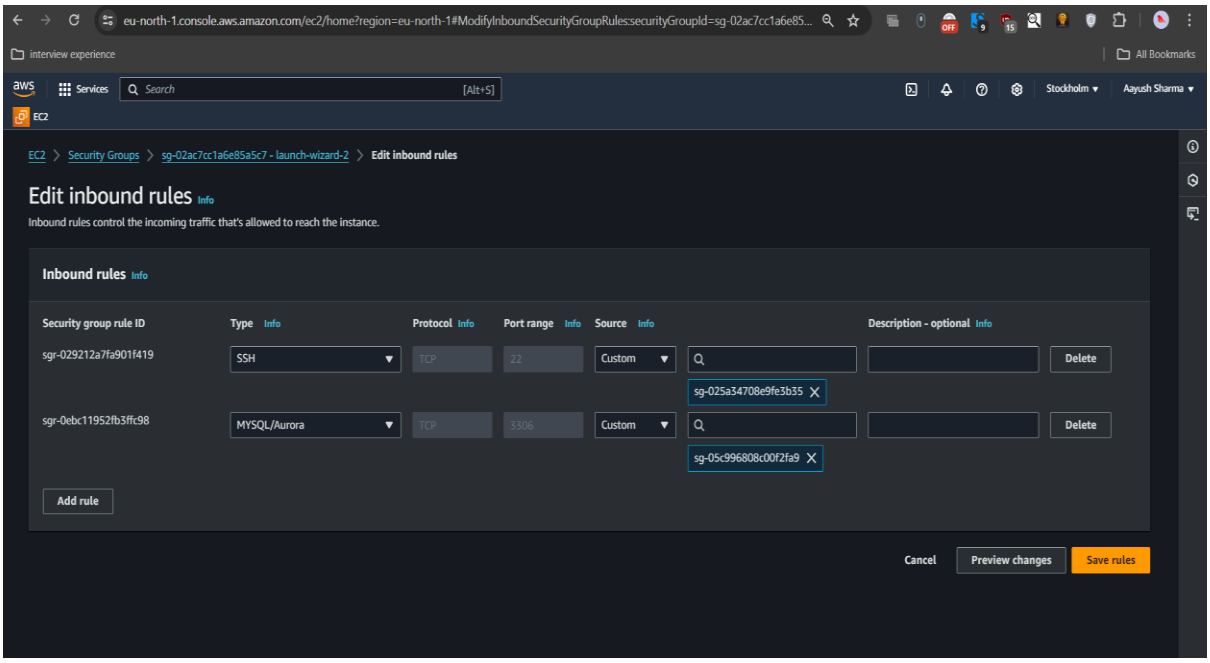
Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Destination Info		Description - optional Info	
sgr-0476c673c9c68a43b	All traffic	All	All	Custom	0.0.0.0/0 X		Delete
sgr-0844402484ba4be57	Custom TCP	TCP	8000	Custom	sg-05c996808c00f2fa9 X		Delete
sgr-0dcb705343c477195	DNS (TCP)	TCP	53	Custom	0.0.0.0/0 X		Delete

Add rule

3.) database instance (sg-02ac7cc1a6e85a5c7)



eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-02ac7cc1a6e85...
interview experience
All Bookmarks
aws
Services
Search
[Alt+S]
Stockholm
Aayush Sharma
EC2

EC2 > Security Groups > sg-02ac7cc1a6e85a5c7 - launch-wizard-2 > Edit inbound rules

Edit inbound rules Info

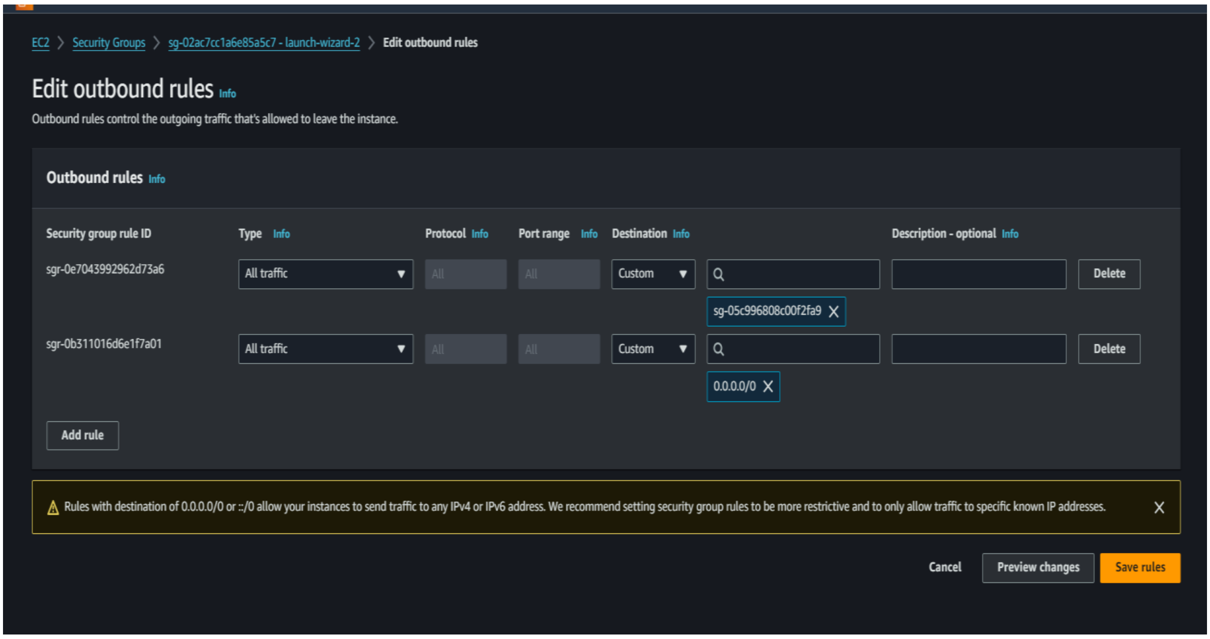
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-029212a7fa901f419	SSH	TCP	22	Custom	sg-025a34708e9fe3b35 X		Delete
sgr-0ebc11952fb3ffc98	MYSQL/Aurora	TCP	3306	Custom	sg-05c996808c00f2fa9 X		Delete

Add rule

Cancel   Preview changes   Save rules


EC2 > Security Groups > sg-02ac7cc1a6e85a5c7 - launch-wizard-2 > Edit outbound rules

Edit outbound rules Info

Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

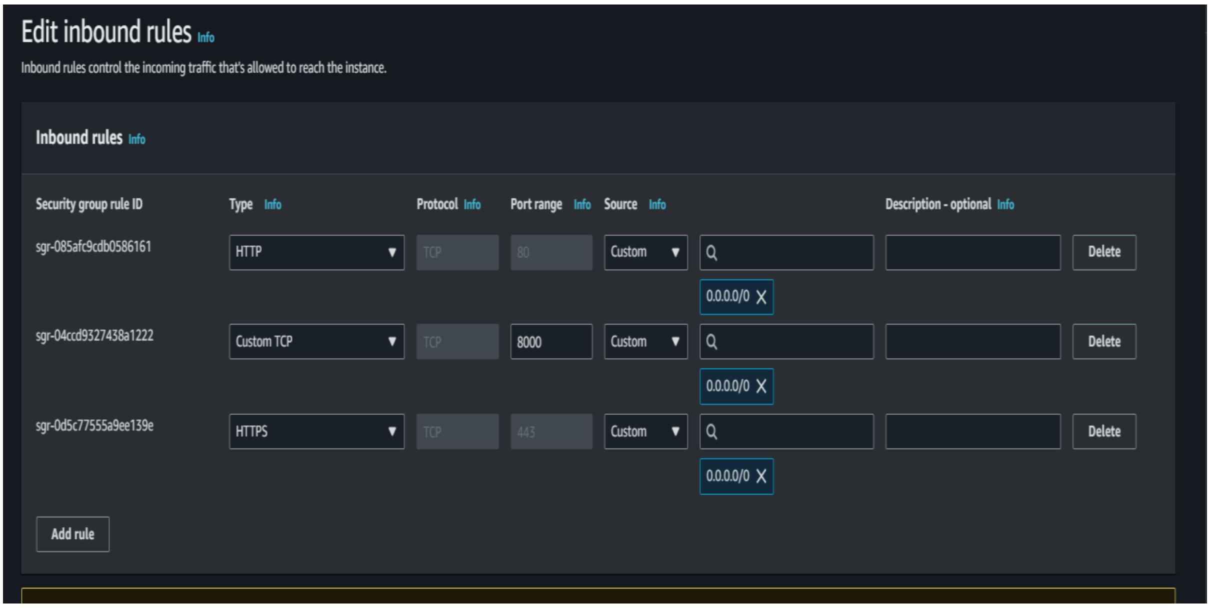
Security group rule ID	Type Info	Protocol Info	Port range Info	Destination Info		Description - optional Info	
sgr-0e7043992962d73a6	All traffic ▼	All	All	Custom ▼	sg-05c996808c00f2fa9 X		Delete
sgr-0b311016d6e1f7a01	All traffic ▼	All	All	Custom ▼	0.0.0.0/0 X		Delete

Add rule

⚠ Rules with destination of 0.0.0.0/0 or ::/0 allow your instances to send traffic to any IPv4 or IPv6 address. We recommend setting security group rules to be more restrictive and to only allow traffic to specific known IP addresses. X

Cancel | Preview changes | Save rules

4.) backend loadbalancer (sg-0319f1c5b53b6d743)

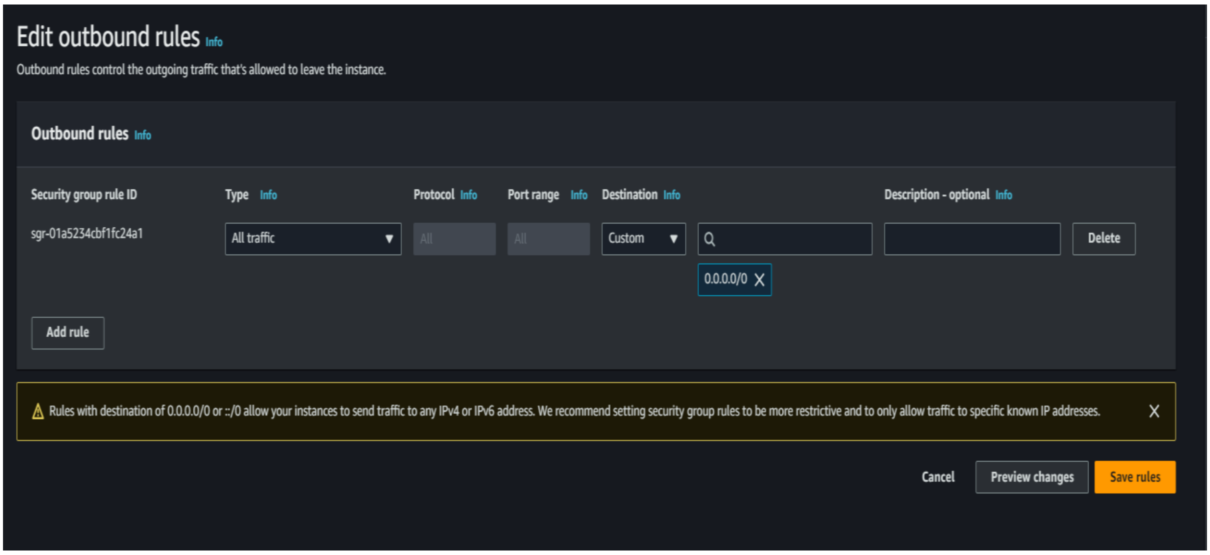

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-085afc9cdb0586161	HTTP ▼	TCP	80	Custom ▼	0.0.0.0/0 X		Delete
sgr-04ccd9327438a1222	Custom TCP ▼	TCP	8000	Custom ▼	0.0.0.0/0 X		Delete
sgr-0d5c77555a9ee139e	HTTPS ▼	TCP	443	Custom ▼	0.0.0.0/0 X		Delete

Add rule



Edit outbound rules Info

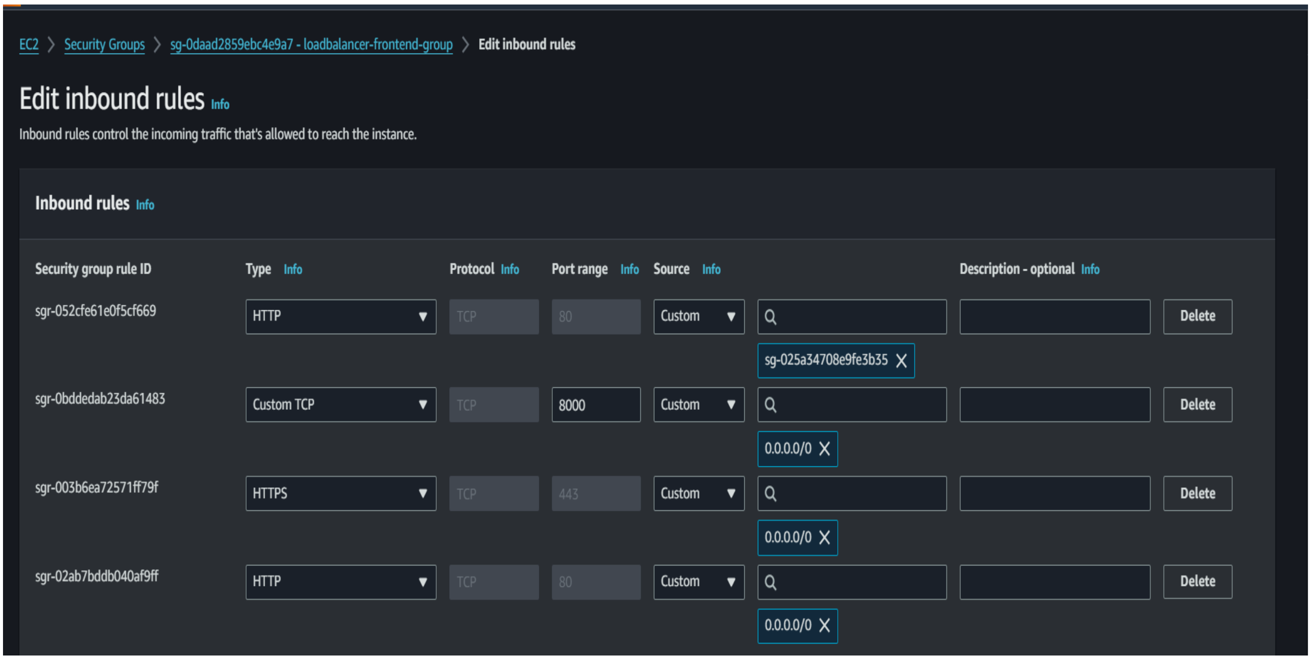
Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

Security group rule ID	Type	Protocol	Port range	Destination		Description - optional
sgr-01a5234cbf1fc24a1	All traffic	All	All	Custom	0.0.0.0/0	

⚠ Rules with destination of 0.0.0.0/0 or ::/0 allow your instances to send traffic to any IPv4 or IPv6 address. We recommend setting security group rules to be more restrictive and to only allow traffic to specific known IP addresses.

5.) frontend load balancer (sg-0daad2859ebc4e9a7)



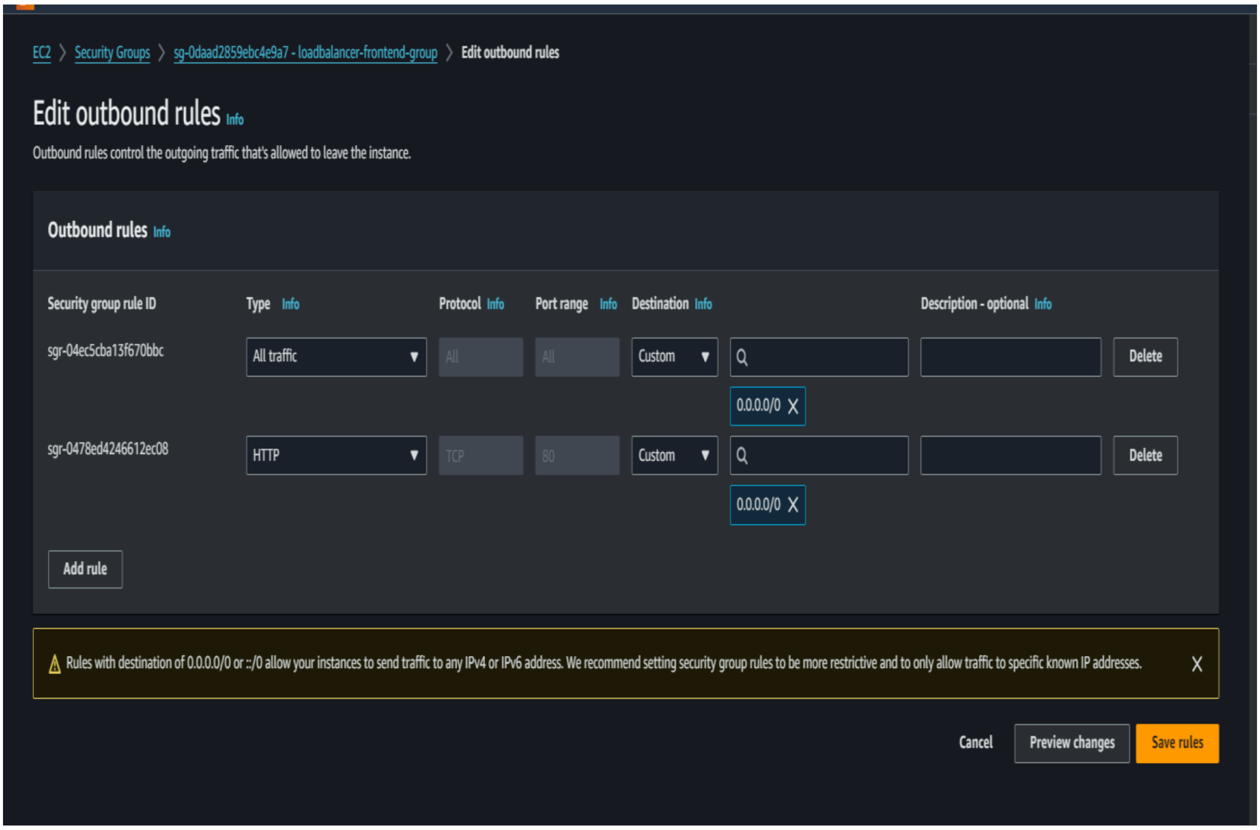
EC2 > Security Groups > sg-0daad2859ebc4e9a7 - loadbalancer-frontend-group > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-052cfe61e0f5cf669	HTTP	TCP	80	Custom	sg-025a34708e9fe3b35	
sgr-0bddedab23da61483	Custom TCP	TCP	8000	Custom	0.0.0.0/0	
sgr-003b6ea72571ff79f	HTTPS	TCP	443	Custom	0.0.0.0/0	
sgr-02ab7bddb040af9ff	HTTP	TCP	80	Custom	0.0.0.0/0	



EC2 > Security Groups > sg-0daad2859ebc4e9a7 - loadbalancer-frontend-group > Edit outbound rules

Edit outbound rules Info

Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Destination Info		Description - optional Info	
sgr-04ec5cba13f670bbc	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-0478ed4246612ec08	HTTP	TCP	80	Custom	0.0.0.0/0		Delete

Add rule

⚠ Rules with destination of 0.0.0.0/0 or ::/0 allow your instances to send traffic to any IPv4 or IPv6 address. We recommend setting security group rules to be more restrictive and to only allow traffic to specific known IP addresses.

Cancel | Preview changes | Save rules

Nginx Configuration

```
upstream backend {
    server internal-backend-lb-354987202.eu-north-1.elb.amazonaws.com;  # AWS internal load
balancer
    # Uncomment and add additional servers if necessary
    # server 10.0.2.88:8000;
}

server {
    listen 80;
    server_name frontend-lb-1116214595.eu-north-1.elb.amazonaws.com;

    # Serve static files
    location /static/ {
        alias /new_chatapp/fundoo/static/;  # Ensure this path is correct for static files
    }

    # Proxy pass backend requests to upstream block
    location / {
        proxy_pass http://backend;  # Reference the upstream block for backend requests
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```