INTERNSHIP PROJECT – CLOUD COMPUTING.

For my internship assignment at Animemangatoon.com, I developed and deployed a cloud-based web application inspired by 'The 05 Best Fantasy Manhwa You Must Read Now.' Utilizing **Amazon Web Services (AWS)**, I ensured that the application was highly available, scalable, and secure. AWS S3 was used for static web hosting, making it cost-effective and efficient for serving content. Additionally, the web app was optimized to handle high traffic using **Auto-Scaling**, providing a global distribution network for faster content delivery.

With AWS's robust suite of services, the project was designed with front-end technologies, avoiding the use of backend frameworks while still ensuring a responsive and reliable experience. The app was version-controlled and deployed via **GitHub**, following cloud best practices for easy deployment and accessibility.

Github link :- Prashant-8113/cloud-assignment - GitHub

Steps for complete site setup and deployment.

1. Project Planning & Research

- Researched the content for the web application based on 'The 05 Best Fantasy Manhwa' You Must Read Now.
- Focused on creating a front-end using HTML, CSS technologies.

2. Setting Up the AWS Environment

- Created an **AWS account** and set up the necessary permissions.
- Planned for a cloud-based deployment using **Amazon EC2**, **S3**, **Elastic Load Balancer**, and **Auto Scaling**.

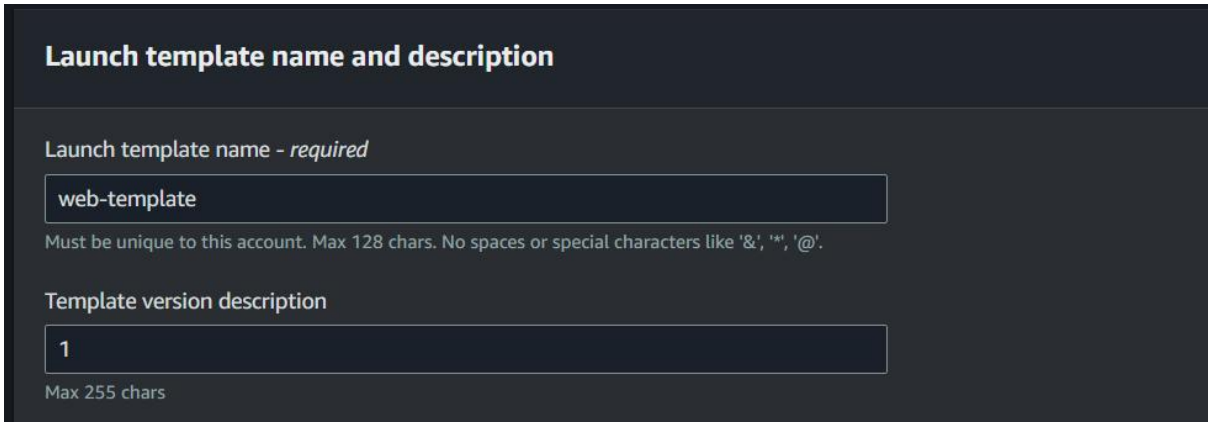
3. Developing the Web Application

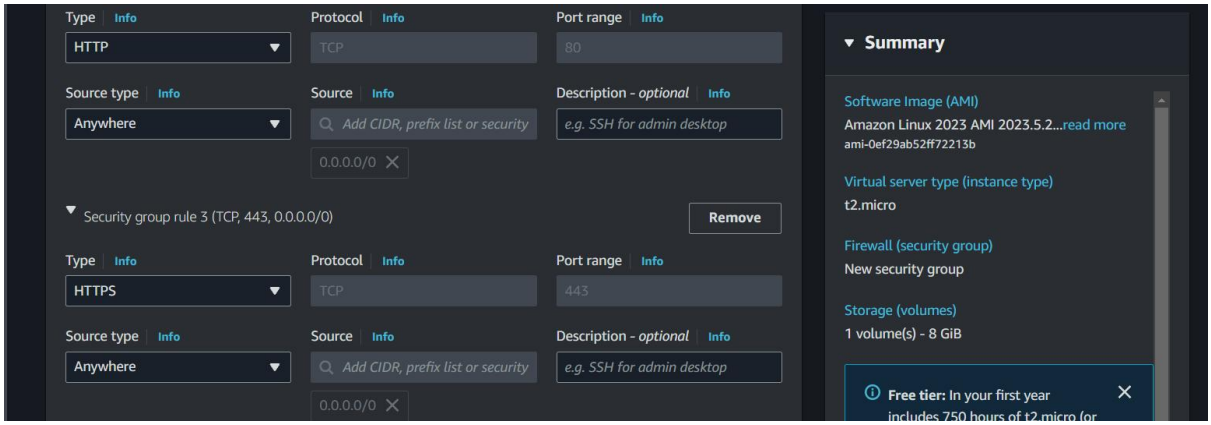
- Built the front-end of the application using HTML, CSS.
- Organized all files (HTML, CSS, images) in preparation for deployment on EC2.
- Site is fulfilled with all the requirements as per animemangatoon.com specified.

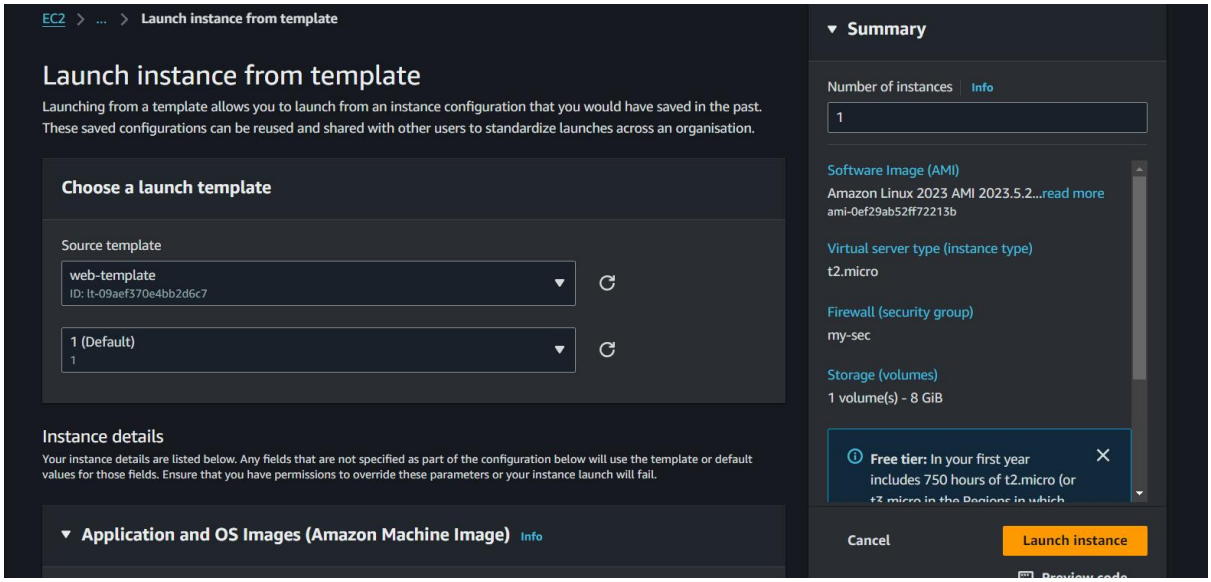
4. Configuring EC2 for Deployment

- Launched an **Amazon EC2** instance (Virtual Machine) to serve as the host for the application.

- Installed a web server (like Apache or Nginx) on the EC2 instance to serve the static web content.
- Uploaded the front-end files (HTML, CSS) to the EC2 instance using **SSH**.
- Allowed HTTP and HTTPS access for web access.





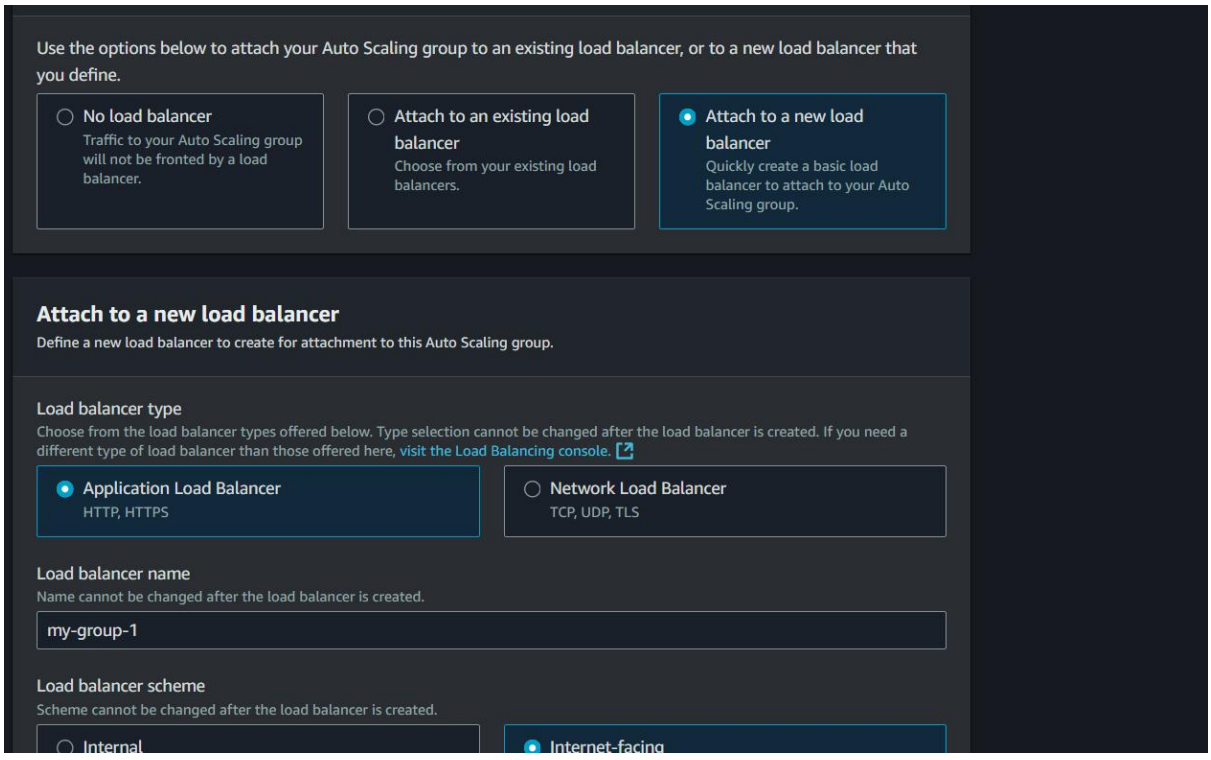


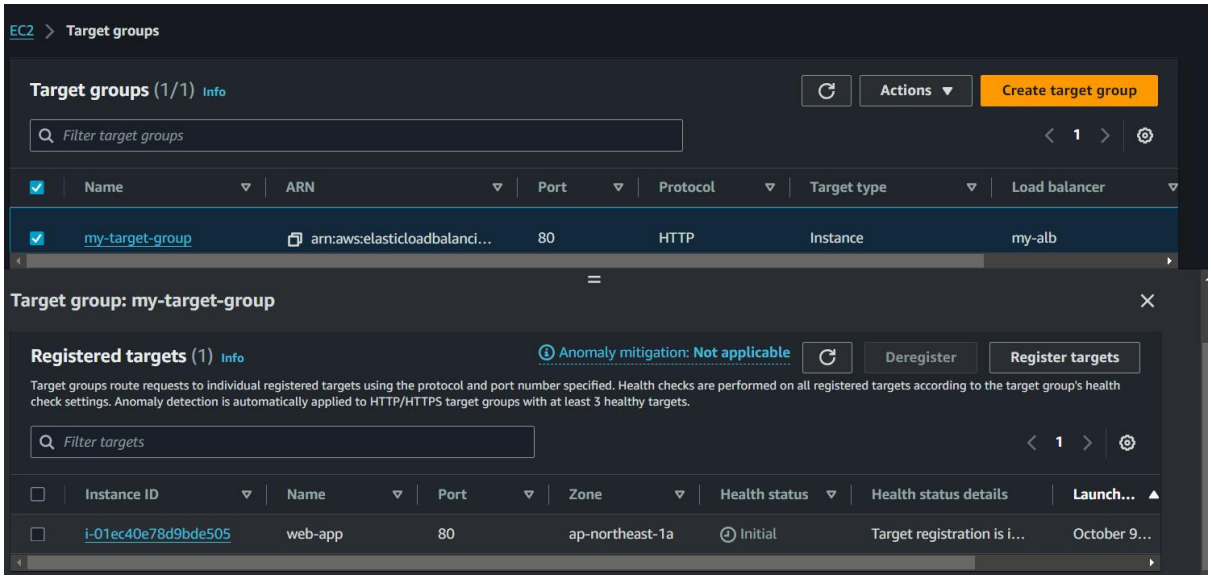
- Code for ec2 deployment.

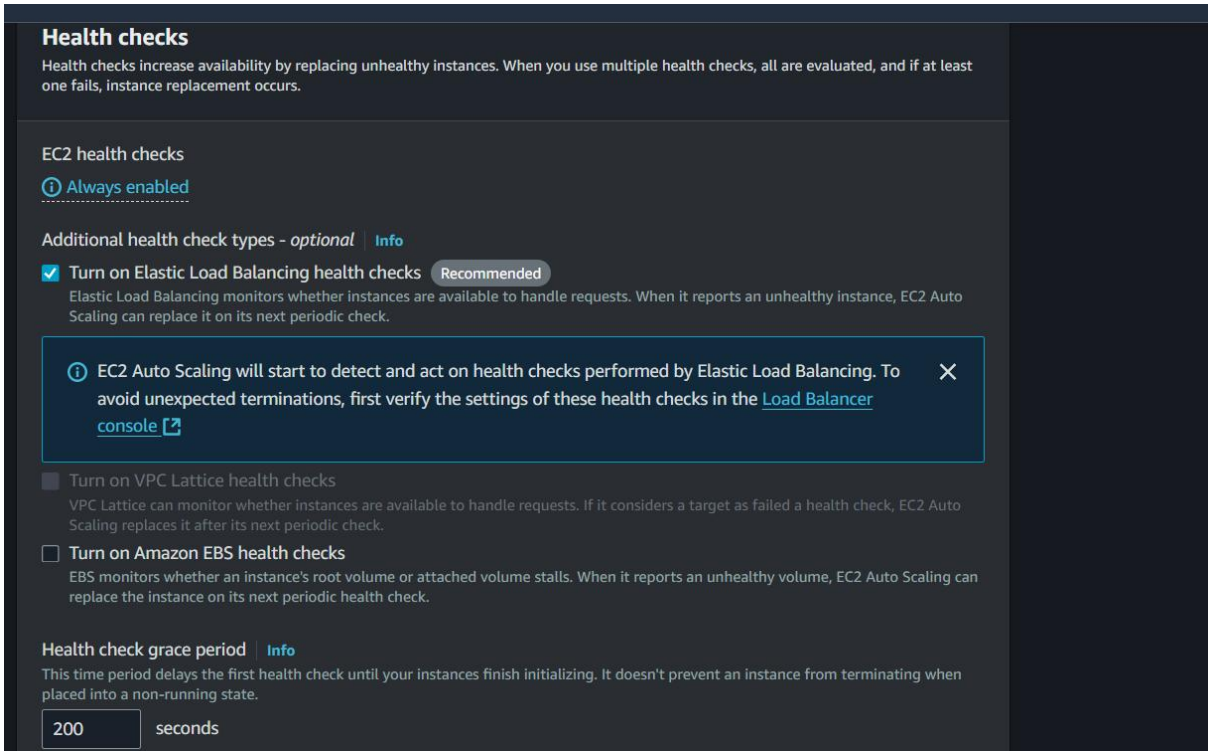
```
[root@ip-172-31-34-196 html]# history
    1  yum install httpd -y
    2  systemctl restart httpd
    3  sudo yum install nginx -y
    4  systemctl start nginx
    5  systemctl enable nginx
    6  yum install git -y
    7  cd /var/www/html/
    8  git clone https://github.com/Prashant-8113/cloud-assignment.git
    9  mv /var/www/html/your-repo/* /var/www/html/
   10  sudo chown -R ec2-user:ec2-user /var/www/html/
   11  sudo chmod -R 755 /var/www/html/
   12  history
[root@ip-172-31-34-196 html]#
```

5. Load Balancer Configuration

- Configured an **Elastic Load Balancer (ELB)** to distribute incoming traffic across multiple EC2 instances, ensuring high availability and fault tolerance.
- Set up the ELB to route requests to your EC2 instance hosting the web application.

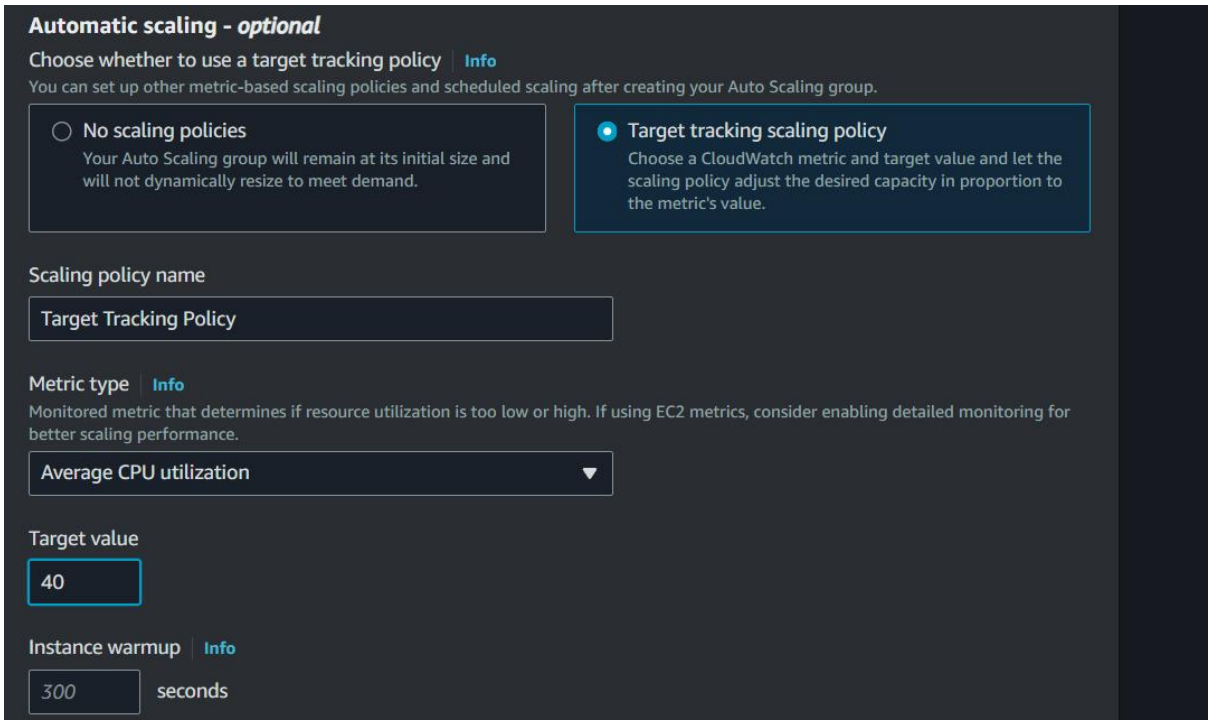



EC2 > Target groups
Target groups (1/1) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
my-target-group
arn:aws:elasticloadbalanci...
80
HTTP
Instance
my-alb
Target group: my-target-group
Registered targets (1) Info
Anomaly mitigation: Not applicable
Deregister
Register targets
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.
Filter targets
Instance ID
Name
Port
Zone
Health status
Health status details
Launch...
i-01ec40e78d9bde505
web-app
80
ap-northeast-1a
Initial
Target registration is i...
October 9...

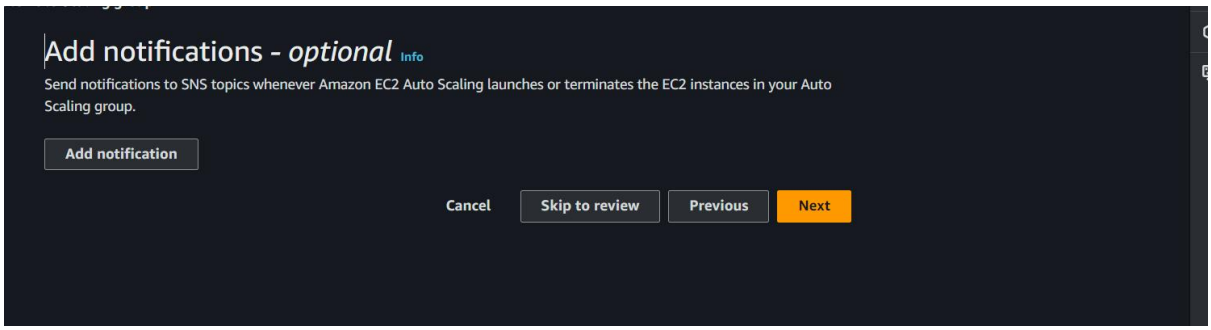

Health checks
Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.
EC2 health checks
Always enabled
Additional health check types - optional Info
Turn on Elastic Load Balancing health checks Recommended
Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.
EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the Load Balancer console
Turn on VPC Lattice health checks
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.
Turn on Amazon EBS health checks
EBS monitors whether an instance's root volume or attached volume stalls. When it reports an unhealthy volume, EC2 Auto Scaling can replace the instance on its next periodic health check.
Health check grace period Info
This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.
200
seconds

6. Enabling Auto Scaling for EC2

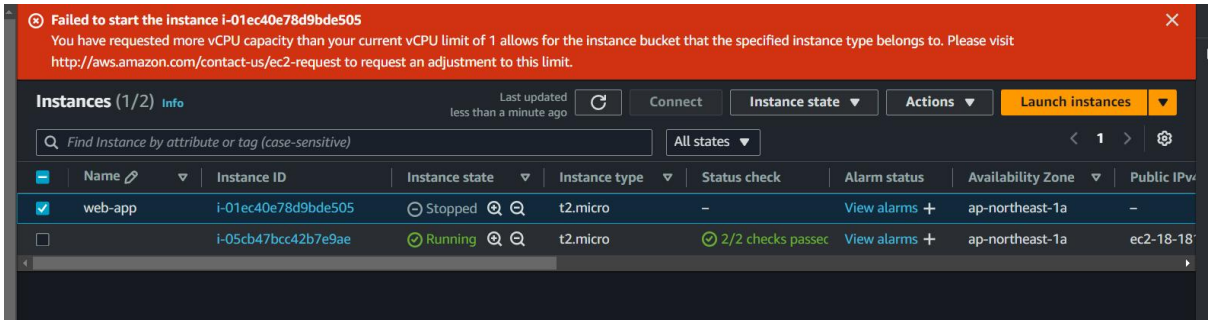
- Configured **AWS Auto Scaling** to automatically add or remove EC2 instances based on traffic demand.
- This ensures that the application can scale dynamically, handling high traffic loads during peak times and reducing costs during low-traffic periods.

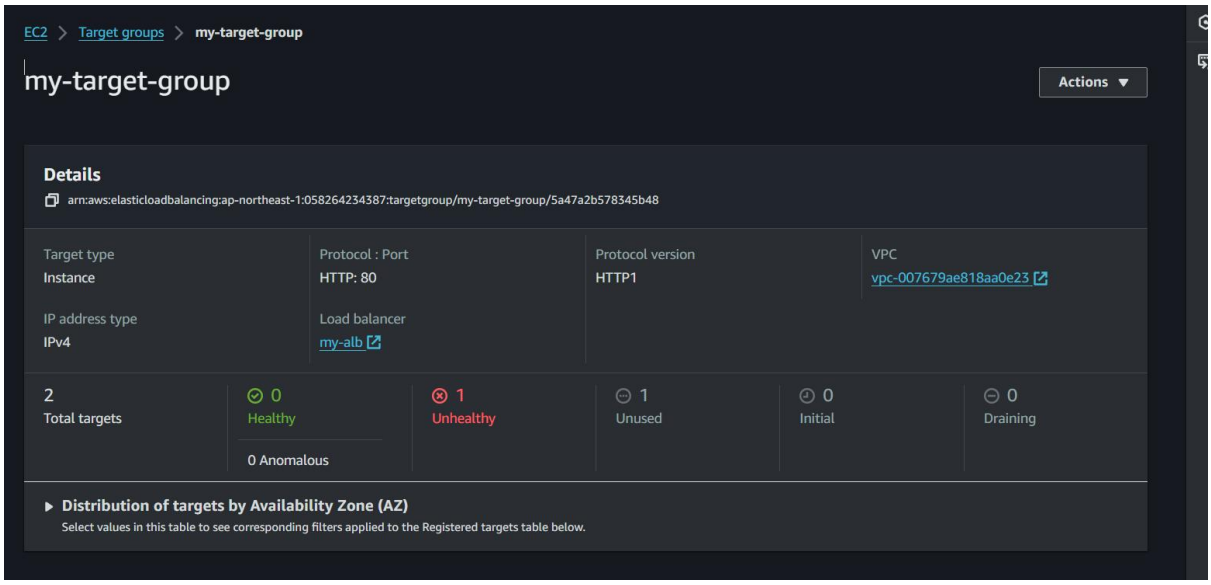


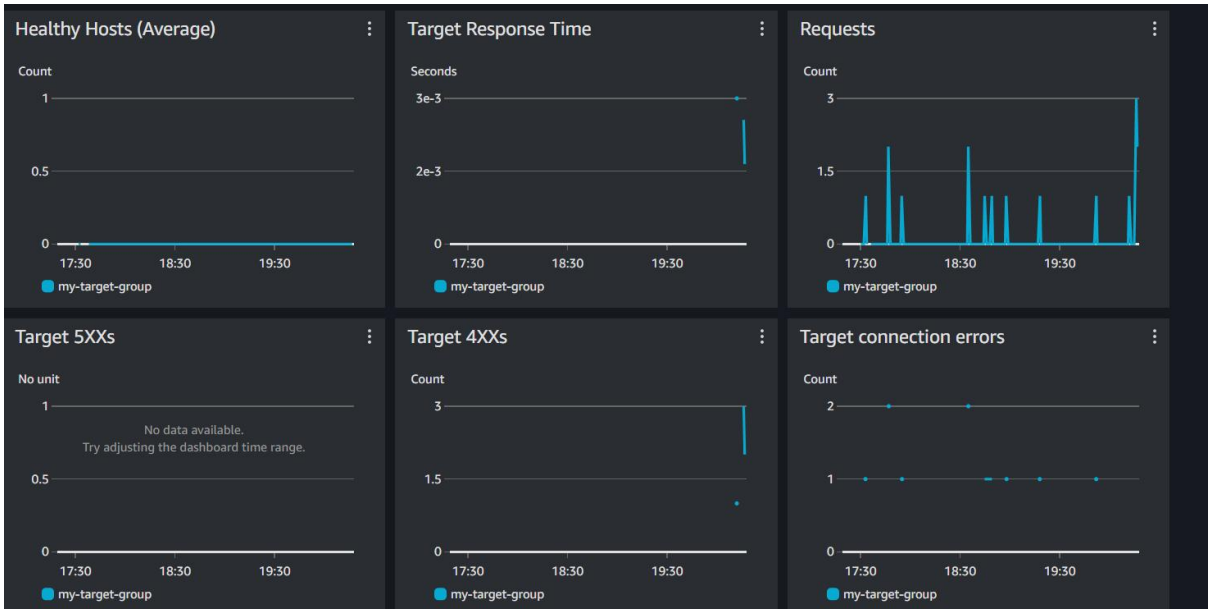
(Optional step for giving sms system for getting real-time information.)



- Real time monitoring and autoscaling.

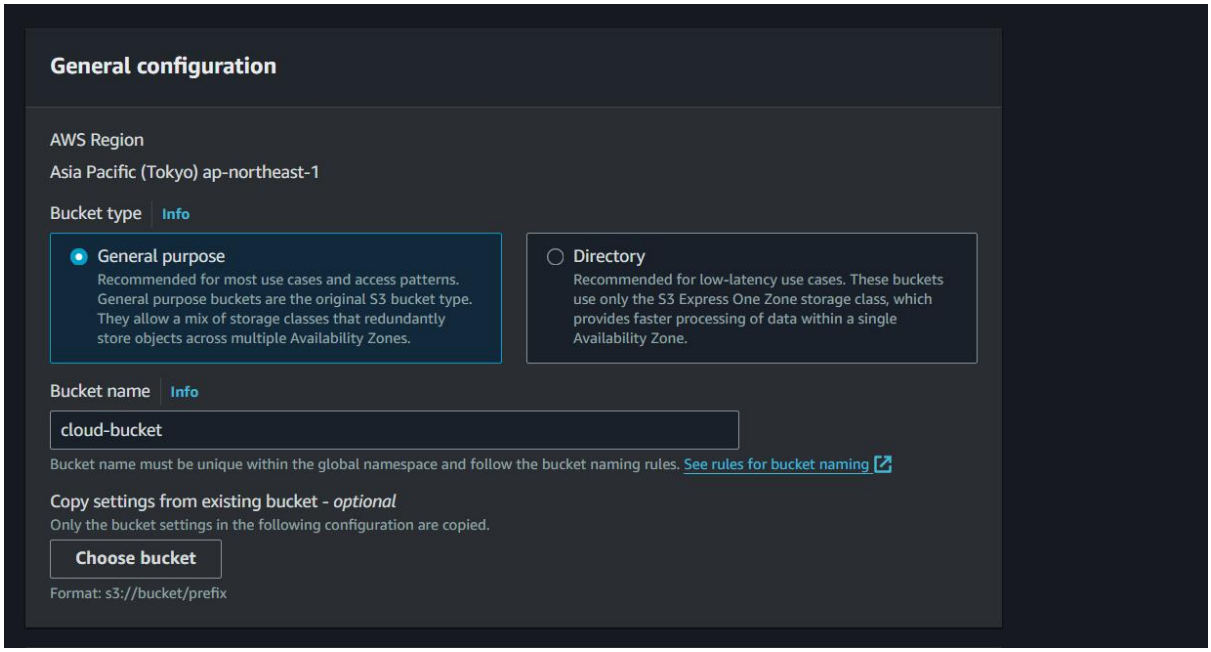


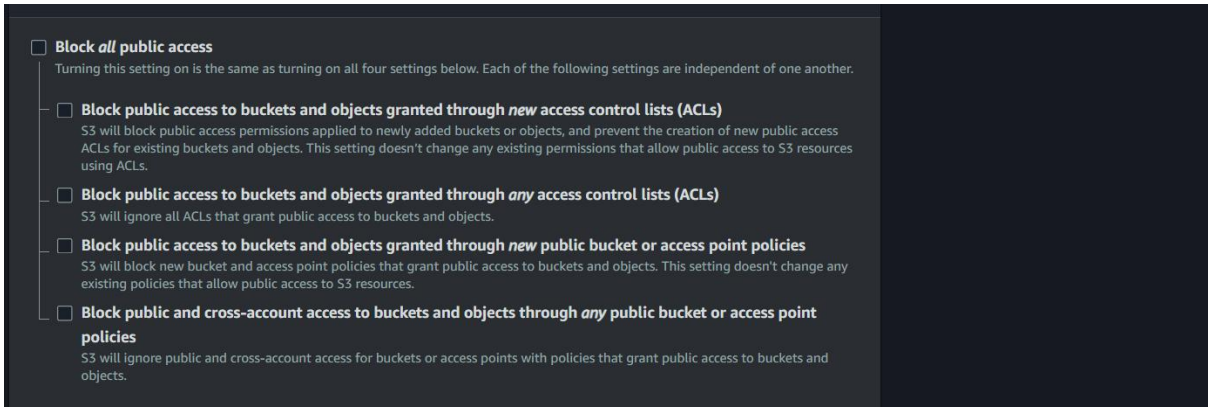


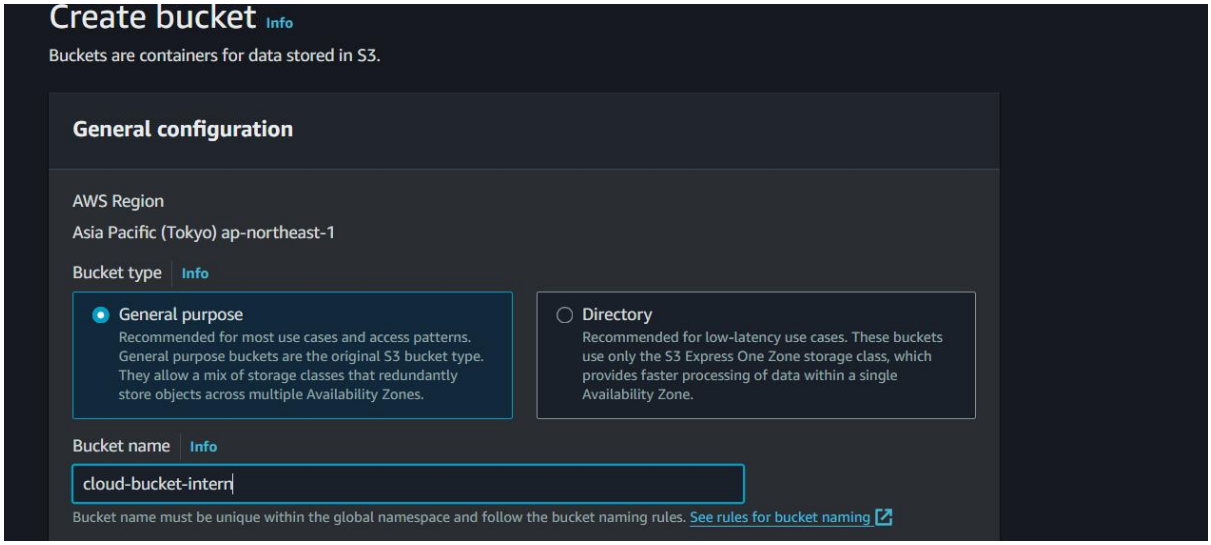


7. Setting Up Amazon S3 for File Storage

- Used **Amazon S3** to store large static assets, such as images, for the website.
- Configured the web server on EC2 to retrieve these assets from the S3 bucket to reduce load on the server and increase performance.



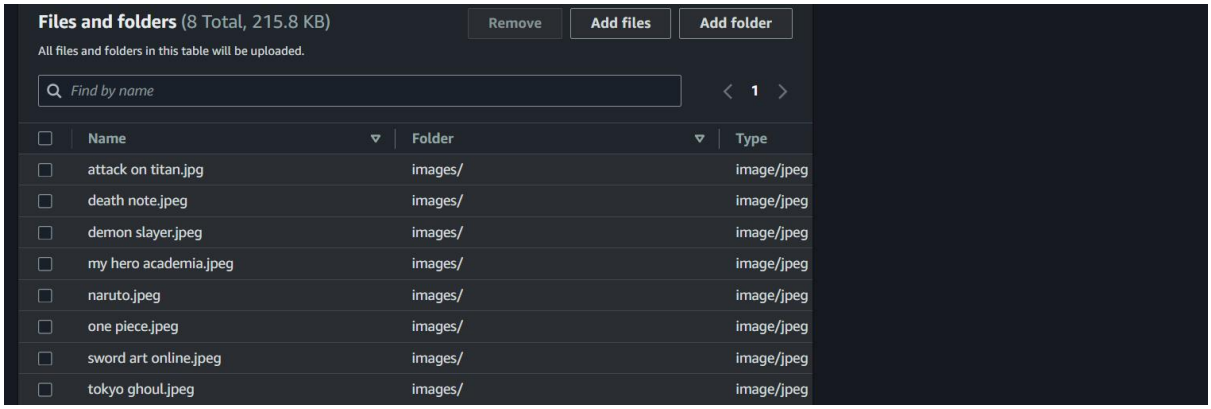




- **Public Access to S3 Bucket:** To enable public read access for all objects in the S3 bucket, configure the bucket's permissions by applying a policy that allows the :- s3:GetObject action to all users (Principal: *).

This ensures that anyone on the internet can access and download files from the bucket.

- All the images and other static code assets added to S3 bucket



8. Version Control with GitHub

- Managed the project's code through **GitHub**, enabling version control and easy collaboration.
- Uploaded the web app to a GitHub repository and linked it for submission.
- Github link :- [Prashant-8113/cloud-assignment - GitHub]

9. Testing & Validation

- Conducted thorough testing of the web application on the EC2 instance, ensuring that it responds properly under different load conditions.
- Verified the performance of the Auto Scaling and Load Balancer configurations.
- Web page link :- http://cloud-bucket-intern.s3-website-ap-northeast-1.amazonaws.com/