Host the Application on AWS

- **Step 1: Launch an EC2 Instance**:
    1. Open the **EC2 Console** and launch an instance using a Linux-based AMI (e.g., Amazon Linux 2).
    2. Choose the instance type (e.g., t2.micro for free tier).
    3. Configure security groups:
        - Allow HTTP (port 80) and HTTPS (port 443).
        - Allow SSH (port 22) for management.
    4. Create and attach a key pair for SSH access.
    5. Install a web server (Apache/Nginx) and deploy your application.

Commands to install and start Apache:

```
sudo yum update -y

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd
```

- **Step 2: Deploy Application Files**:
    1. Upload your web files (HTML, CSS, JS, and JSON data) to the EC2 instance.
    2. Place them in the web server directory (e.g., /var/www/html/).

3. Store Images on S3

- **Step 1: Create an S3 Bucket**:
    1. Open the **S3 Console** and create a new bucket (e.g., manhwa-images-bucket).
    2. Upload the manhwa cover images.
    3. Enable public access for the images by configuring the bucket policy.

- **Step 2: Use S3 URL in Application**:
    1. In your web app, use the S3 object URLs for each image.

4. Set Up RDS for Database

- **Step 1: Create an RDS Instance**:
    1. Open the **RDS Console** and create an RDS instance (MySQL/PostgreSQL).
    2. Choose the free tier option (e.g., db.t2.micro).
    3. Create a database (e.g., manhwa_db) and a table to store manhwa data.

Example table structure:

sql

```
CREATE TABLE manhwa (
  id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255),
  genre VARCHAR(255),
  description TEXT
);
```

5. Set Up Auto-Scaling

- **Step 1: Configure Auto-Scaling Group**:
    1. Go to **EC2 Auto Scaling** and create a new Auto Scaling group.
    2. Set the minimum number of instances (e.g., 1) and a maximum (e.g., 3).
- **Step 2: Set Scaling Policies**:
    1. Create scaling triggers based on CPU utilization (e.g., if CPU > 60% for 5 minutes, add a new instance).
    2. Configure a cooldown period and scaling back when the CPU drops.

6. Enable HTTPS with AWS Certificate Manager (ACM)

- **Step 1: Request a Certificate**:
    1. Go to **AWS Certificate Manager** and request a public certificate for your domain (or use a free domain).
- **Step 2: Attach Certificate to Load Balancer**:
    1. Set up an **Application Load Balancer (ALB)** and attach the ACM certificate to enable HTTPS.
    2. Redirect all HTTP traffic (port 80) to HTTPS (port 443).

7. Implement Security Measures

- **Step 1: Configure Security Groups**:
    1. Restrict inbound rules to allow only HTTP (80), HTTPS (443), and SSH (22).
    2. Restrict outbound rules to only allow access to necessary services like S3 and RDS.
- **Step 2: Enable a Firewall (NACL)**:
    1. Configure a Network Access Control List (NACL) to restrict access based on IP ranges if needed.

8. Document Auto-Scaling Configuration

- Explain how you configured auto-scaling:
    - Set a threshold of **60% CPU** for scaling up.
    - Decrease instances when CPU utilization drops below **30%**.
    - Use **CloudWatch** alarms to monitor CPU and trigger scaling actions.

9. Deploy and Test

- Deploy the web application, database, and S3 integration.
- Test auto-scaling by running a load test (using **Apache JMeter** or a similar tool).