

# **DDoS Protection System for Cloud: Architecture and Tools**

Research Project  
Sponsored by  
Silver Oak University

**Submitted by**  
**Rohit Navin Kandpal**  
**Computer Science & Technology with**  
**Specialization in Cyber Security**  
**Silver Oak College of Engineering and Technology**



## **Silver Oak University**

## INDEX

<b>Sr NO</b>	<b>Content</b>	<b>Page No</b>
01	Abstract	4
02	Introduction	5
03	Objective(s)	6
04	Methodology	7
05	Project Outcome(s)	13
06	Reference(s)	14

## **(A) Technical Details of Project**

### **1. Abstract**

### **2. Introduction**

### **3. Objective**

### **4. Methodology**

- **Requirements Analysis:**
- **System Design:**
- **Hardware Implementation:**
- **Algorithm Development:**
- **Software Development:**
- **Integration and Testing:**
- **Iterative Improvement:**
- **Photos**

### **5. Outcomes**

### **6. References**

## **Abstract**

This project focuses on the development of a Distributed Denial-of-Service (DDoS) Protection System for cloud-hosted web applications. The rising use of cloud infrastructure has led to an increased risk of DDoS attacks, which can severely disrupt the availability of online services by overwhelming the server with malicious traffic.

To counteract this threat, the system designed in this project monitors incoming traffic, detects suspicious patterns, and blocks malicious requests using advanced techniques, including request throttling and IP-based filtering. Utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js), AWS EC2 instances for cloud infrastructure, and a proxy server for additional security, this project demonstrates a scalable, cost-effective approach to protecting cloud-hosted applications from DDoS attacks. The system was tested using simulated DDoS attacks to ensure its effectiveness in maintaining service availability. This project presents a robust and practical solution to one of the most common threats in cybersecurity.

## **Introduction**

As more organizations transition their operations to cloud-based platforms, ensuring the availability and security of web applications has become a critical concern. Among the myriad of threats that cloud-hosted services face, Distributed Denial-of-Service (DDoS) attacks remain one of the most disruptive and challenging to mitigate. A DDoS attack typically involves flooding a target server with an overwhelming volume of traffic from multiple sources, rendering the service inaccessible to legitimate users.

This project aims to address the challenges posed by DDoS attacks by developing a protective system that can detect and mitigate such attacks in real-time. By leveraging cloud services, modern web technologies, and automated traffic analysis tools, the DDoS Protection System provides a proactive defense mechanism for cloud-hosted web applications.

The system not only monitors and filters incoming traffic to detect suspicious activity but also employs automated mitigation strategies to block potential threats while minimizing downtime and service disruption. The project makes use of AWS EC2 for scalable hosting, MongoDB for traffic logging, and proxy servers for an added layer of security, ensuring a comprehensive approach to DDoS prevention.

## **Objective**

The primary objective of this project is to develop a robust DDoS Protection System capable of detecting, mitigating, and preventing DDoS attacks on cloud-hosted applications. Specifically, the project aims to:

**Monitor Web Traffic:** Continuously monitor incoming requests to the server, logging traffic patterns and identifying unusual spikes in activity.

**Detect Suspicious Behavior:** Implement algorithms to detect traffic patterns characteristic of a DDoS attack, such as rapid request rates from specific IP addresses or unusually high traffic volumes.

**Automate Mitigation:** Automatically block IP addresses that are deemed to be suspicious or malicious, preventing them from overwhelming the server.

**Ensure High Availability:** Maintain the availability of the web service by quickly identifying and mitigating DDoS attacks, ensuring minimal disruption to legitimate users.

**Leverage Cloud Infrastructure:** Utilize AWS EC2 instances for flexible and scalable hosting that can handle fluctuations in traffic and support the deployment of the DDoS protection system.

**Incorporate Proxy Servers:** Use proxy servers to provide an additional layer of protection by acting as intermediaries between the client and the main application server, preventing direct access by malicious users.

**Test the System:** Simulate various types of DDoS attacks to test the system's effectiveness, ensuring that it can reliably detect and mitigate attacks under different conditions.

## Methodology

### Requirement Analysis

In the planning phase, a comprehensive requirements analysis was conducted to identify the technical components needed for the project. The following requirements were established:

**Cloud Hosting:** AWS EC2 instances were chosen for cloud hosting due to their scalability and flexibility. EC2 allows the application to handle traffic spikes by adjusting resources dynamically, making it well-suited for DDoS protection.

**Traffic Monitoring:** MongoDB was selected as the database to log incoming traffic requests. The database stores detailed information about each request, including the IP address, request path, and timestamp, which can later be analyzed to detect malicious activity.

**DDoS Detection and Mitigation:** The system needs to employ an algorithm that monitors traffic for abnormal patterns, such as a high volume of requests from a single IP in a short period. Once suspicious activity is detected, the system should automatically block the offending IP addresses.

**Frontend Landing Page:** A simple user-friendly landing page to showcase the architecture and behind the page backend will continue to run. This will be built using React.js.

**Proxy Server:** An intermediary proxy server (such as Nginx) is necessary to add a layer of security by filtering traffic before it reaches the main application server.

**Testing Tools:** Tools such as **LOIC** (Low Orbit Ion Cannon) and **hping3** will be used to simulate DDoS attacks and test the system's ability to detect and mitigate them.

## System Design

The system is designed to operate as a multi-layered architecture, with each component playing a specific role in the DDoS detection and mitigation process:

**AWS EC2 Instances:** The core of the system is hosted on AWS EC2 instances, which provide the computational resources needed to handle large volumes of incoming traffic. EC2 offers scalability, meaning that as traffic increases, additional resources can be allocated automatically to maintain system performance.

**Proxy Server:** A proxy server, implemented using **Nginx**, is deployed between the client and the main server to act as a traffic filter. The proxy server intercepts all incoming requests, blocking malicious traffic before it can reach the backend, thus reducing the load on the main server.

**Traffic Monitoring & Logging:** Every incoming request to the server is logged into **MongoDB**. Each log entry contains the IP address of the requester, the request path, the timestamp, and other metadata that is used to analyze traffic patterns.

**DDoS Detection Algorithm:** A threshold-based detection algorithm monitors the number of requests made by each IP address over a given period. If an IP address exceeds the request threshold (e.g., more than 10 requests per second), it is flagged as suspicious. The system automatically blocks the IP for a predefined period, preventing further requests.

**Frontend Dashboard:** It will be built using React so that by using `useEffect` hook to simultaneously get the logs data in the background without refreshing the webpage, which is the main feature of React.

**Cloudflare Integration (Optional):** To further enhance security, services like **Cloudflare** can be integrated for additional features such as rate limiting, Web Application Firewall (WAF), and SSL encryption.



## Hardware Implementation

No specialized hardware is required for this project since the system is hosted entirely in the cloud. The project utilizes **AWS EC2 instances** to host the application and handle traffic, which ensures flexibility and scalability. The proxy server and DDoS protection algorithms run on these virtualized instances. This cloud-based infrastructure eliminates the need for physical hardware, making the system cost-effective and easier to scale based on traffic demands.

## Algorithm Development

The DDoS detection algorithm forms the core of this project. Its primary function is to monitor traffic patterns and identify potential attacks:

**Request Thresholds:** The algorithm monitors the number of requests each IP address makes within a set time frame. If an IP exceeds a certain number of requests (e.g., 10 requests in 10 seconds), it is flagged as suspicious.

**Blocking Mechanism:** Once an IP is flagged, the system blocks subsequent requests from that IP for a specific period. The blocked IPs are stored in a temporary blacklist that expires after a predefined time.

**Adaptive Learning:** In future iterations, the algorithm could be enhanced to include machine learning techniques that adapt to traffic patterns over time, making the system more intelligent and capable of detecting more complex attack patterns.

## Software Development

The software development of this project is divided into two primary areas: backend and frontend.

### Backend:

- The backend is built using **Node.js** with **Express.js** as the framework to handle HTTP requests. It is responsible for routing incoming traffic, logging the traffic data into MongoDB, and implementing the DDoS detection logic.
- **MongoDB** is used to store and manage the traffic logs. This database enables real-time analysis of traffic data, allowing the system to identify patterns that may indicate a DDoS attack.
- **AWS EC2** hosts the backend services, ensuring that the system remains accessible and can scale as needed.

### Frontend:

- The frontend is developed using **React.js**. It provides administrators with a clean, responsive user interface to monitor incoming traffic, view flagged IPs, and configure protection settings.
- **Axios** is used to fetch data from the backend and display real-time traffic logs.

### Proxy Server:

- **Nginx** acts as the proxy server, filtering incoming traffic and blocking malicious requests before they reach the main application server.

### DDoS Testing Tools:

- The system is tested using tools like **LOIC** and **hping3** to simulate both DoS and DDoS attacks. These tools generate large amounts of traffic, allowing us to assess the system's response to high-traffic situations.

## Integration and Testing

Integration and testing are critical components of this project. After the individual components (backend, frontend, proxy, etc.) were developed, they were integrated into a single system and tested under various scenarios:

**DDoS Simulation:** Tools like **LOIC** and **hping3** were used to simulate DoS and DDoS attacks. These tools helped test the system's ability to detect and block malicious traffic without affecting legitimate users.

**Performance Testing:** Using tools like **Apache JMeter**, we simulated legitimate traffic to assess the system's performance under normal conditions. This helped ensure that the DDoS detection mechanism does not block legitimate traffic during high-traffic periods.

**Integration Testing:** The system was tested as a whole, ensuring that all components—AWS infrastructure, proxy server, backend logic, and frontend monitoring—worked together seamlessly. This helped identify and fix any bottlenecks or integration issues.

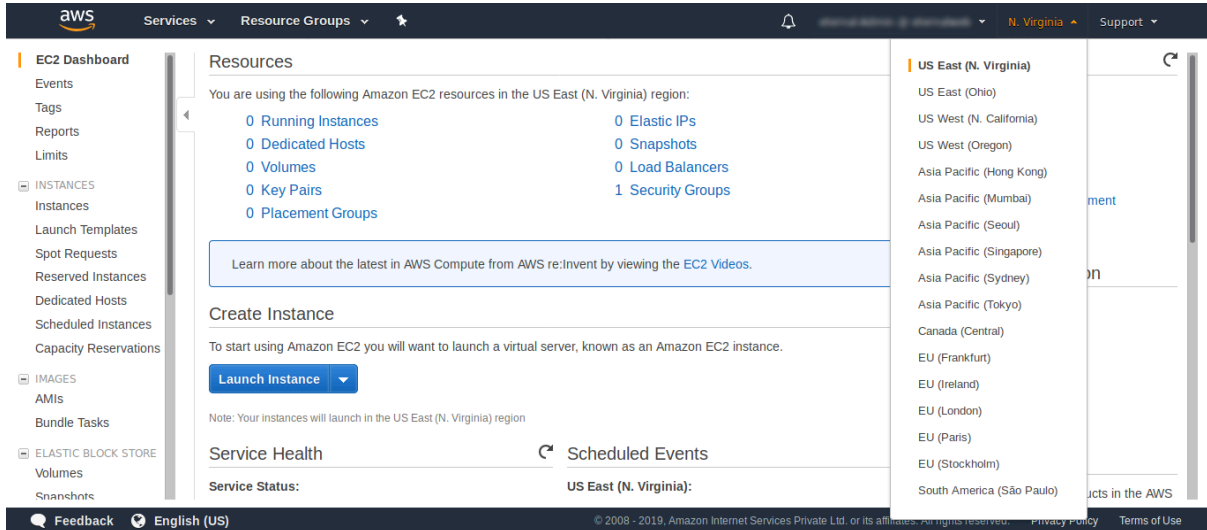
## Iterative Improvement

As part of the development process, several iterative improvements were made:

1. **Threshold Adjustment:** During initial testing, the detection algorithm's request thresholds were found to be too low, resulting in false positives (legitimate users being blocked). The thresholds were fine-tuned to strike a balance between security and usability.
2. **Frontend Enhancements:** Based on feedback, the frontend was improved to offer more detailed insights into traffic data and provide better visualization tools for administrators.
3. **Database Optimization:** As traffic logs grew in size, performance optimizations were made to the MongoDB queries to handle larger datasets more efficiently.

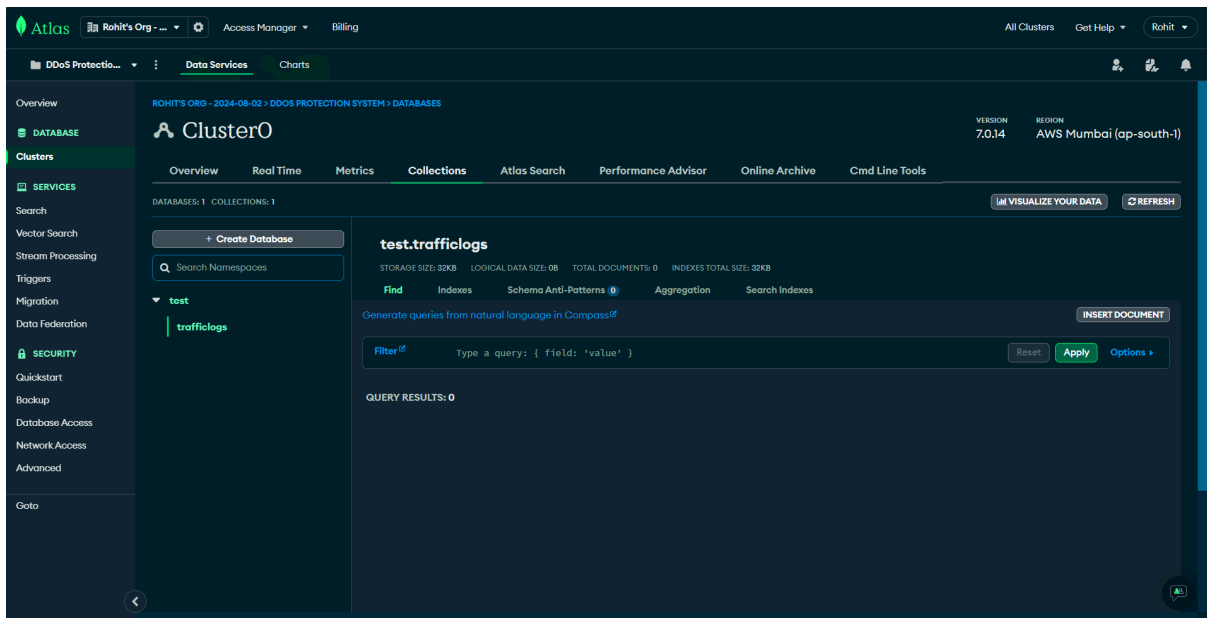
## Photos

## AWS E2C Setup



The screenshot displays the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a search bar. The left-hand navigation pane lists various AWS services, with 'EC2 Dashboard' selected. The main content area is titled 'Resources' and shows the current region as 'US East (N. Virginia)'. It lists the following resources: 0 Running Instances, 0 Elastic IPs, 0 Snapshots, 0 Volumes, 0 Load Balancers, 0 Key Pairs, and 1 Security Groups. Below this list is a link to 'Learn more about the latest in AWS Compute from AWS re:Invent by viewing the EC2 Videos.' The 'Create Instance' section provides instructions on how to launch an Amazon EC2 instance and includes a 'Launch Instance' button. The right-hand navigation pane shows a list of regions, with 'US East (N. Virginia)' highlighted.

## MongoDB Logs Database



The screenshot shows the MongoDB Atlas web interface. The top navigation bar includes the Atlas logo, 'Rohit's Org', 'Access Manager', and 'Billing'. The left-hand navigation pane lists various services, with 'DATABASE' selected. The main content area is titled 'Cluster0' and shows the 'test.trafficlogs' collection. It includes a search bar, a 'Filter' button, and a 'QUERY RESULTS: 0' section. The right-hand navigation pane shows the 'test.trafficlogs' collection details, including storage size, logical data size, total documents, and index total size.

## **Outcomes**

The project is currently at a very naive state but it successfully tries to implement a DDoS Protection System capable of identifying and mitigating DDoS attacks in real-time. Through the use of AWS EC2 for scalable hosting, MongoDB for real-time traffic analysis, and a proxy server for enhanced security, the system demonstrated its ability to maintain high availability during simulated DDoS attacks.

Key outcomes include:

- Effective detection and blocking of suspicious IP addresses, mitigating DDoS attacks before they could overwhelm the server.
- Successful integration of cloud infrastructure (AWS) with a proxy server, ensuring a scalable and secure hosting environment.
- A user-friendly React-based dashboard for monitoring traffic and managing DDoS protection settings.
- Simulated DDoS attacks were detected and mitigated without significant impact on the availability of the service.

## **Reference**

Nginx Proxy Server: <https://docs.nginx.com/>

React.js Documentation: <https://reactjs.org/>

Express.js Documentation: <https://expressjs.com/>

MongoDB Documentation: <https://docs.mongodb.com/>

DDoS Attack Info at:

<https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>

Architecture Reference:

<https://www.akamai.com/resources/reference-architecture/ddos-protection>

AWS Instance Setup:

[https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2\\_GetStarted.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html)