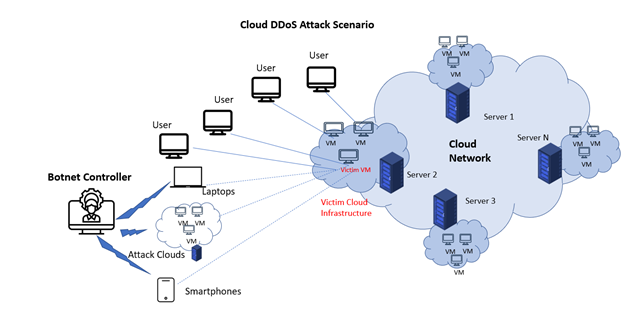
Providing the access of hardware or software as a service to the end user for specific time duration over network, without actually having them physically is known as cloud computing. This includes collections of virtual machines, virtual networks, customized instance of the computer to provide services such as operating system, auto scaling, load balancer and many more as per our need. Adding all this it becomes cloud computing system. To configure the cloud computing environment is easy compared to virtualization on the other hand. Cloud computing comes with three main service layer. Infrastructure as a service (IaaS), which provide infrastructure to the end user like network, bandwidth and storage. Platform as a Service (PaaS), providing platform to the cloud computing users like operating system, applications and any platforms they might need to develop and create application just like database. Software as a service (SaaS) give access of a software to the end user.

DDoS attack is one of the biggest security risk for a cloud computing environment, because here cloud computing resources are shared to multiple users. So, by flooding this system resources with enormous amounts of malicious traffic, one can restrict the availability of that cloud resource to all the end users with whom the access was shared, making it inaccessible. DDoS attacks are intended to increase the utilization of the resources such as memory, network bandwidth and CPU processing in order to make them unavailable to end users by blocking network traffic or denying access to services. It's difficult to detect the difference between the attacker's demands and genuine user requests, especially if the latter come from a very large number of resources and distributed workstations, while dealing with multiple DDoS attacks at all levels in cloud computing systems.

We'll test DDoS attacks on cloud computing services and mitigate the DDoS attacks.

The major focus of this project is on perform DDOS on cloud computing resources for a certain service, but it is not restricted to that.

1. Various Denial of Service assaults are being carried out.
2. Flooding the system with traffic, causing legitimate users to be denied service.
3. Disruption of communication between two machines, restricting access to a service.
4. denying access to a service to a certain system or user.
5. Maintain a botnet network that can be utilized to carry out the DDOS

****

*Figure 4.1 Process Model of Cloud DDOS Attack*

### **List of Major Activities**

Initial project reconnaissance and planning.

Creating a Web Application to be hosted on Cloud Platform.

Creating a Cloud Instance for the Web Application.

Host the Web Application on the Cloud Instance created.

Perform DOS Attacks on the Cloud Service (Website).

Feasible study and Setting up Botnet Server.

Payload creation.

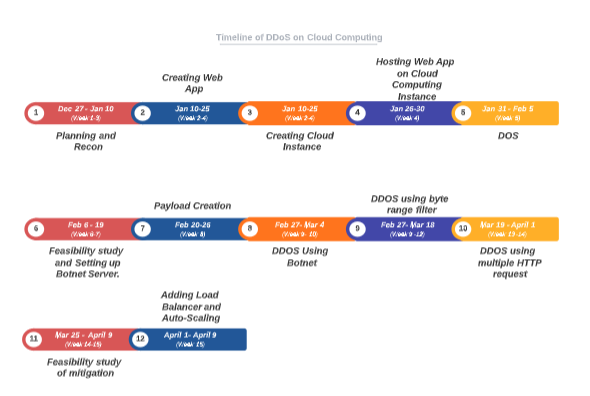
Performing DDOS Attacks using Botnet server.

Using range filter vulnerability for DDOS.

Using multiple HTTP request for DDOS.

Feasibility Study of mitigation.

Adding Load Balancer and Auto-Scaling.



### **Initial project reconnaissance and planning.**

# FLOWCHART:

# 

# DDOS:

* It’s a malicious attempt to disrupt the services (Application, Networking, Infrastructure).
* Disruption could be caused by any of following but not limited to:
  + - Application Stress.
    - Networking Congestion.
    - Connection State Exhaustion.
* While in DDOS multiple resources are focused to disrupt a particular service.

# STEPS:

1. Create a Simple Web Application.
   * 1. Creating an Ecommerce website, which will be used to sell Electronic devices online.
2. Create a Cloud Instance(EC2).
   * 1. Using the AWS console, create a EC2 instance.
     2. Select AMI.
     3. Select type of instance.
     4. Configure the instance.
     5. Add storage to the instance.
     6. Add proper tags.
     7. Configuring the security group.
     8. Verify the details and launch the instance.
3. Host the Web application on Cloud Computing Service.
   * 1. Connect to the EC2 instance using SSH and Install appace Web Server
     2. Change directory to /var/www/ and move the web application.
4. Simple Load testing for the server / Monitoring of Cloud Computing Resources
   * 1. AWS Distributed Load Testing is a service that automates the testing of software applications at scale and under load in order to uncover possible performance issues before they are released.
     2. Without the need for servers, it produces and mimics thousands of connected users that generate transactional records at a steady rate.
5. Performing various DDOS attacks.
   * 1. Flooding system traffic which leads to service denial to legitimate users.
     2. Connection disruption between of two end-devices, so eventually districting access to a service.
     3. Isolating a particular system or user from accessing a service.
6. Checking that the attack is successful.
   * 1. Check Metrics and Log of the EC2 instance
7. DDOS Mitigation.
   * 1. Increase the bandwidth.
     2. Use AWS Shield / Shield Advanced / WAF.
     3. Request redirecting.
        1. Create an S3 bucket that redirects HTTP traffic. This redirects queries to your www subdomain from your zone apex.
        2. Make a CloudFront web distribution and configure it. In front of my S3 web redirect, I employ a CloudFront distribution to take advantage of CloudFront's excellent DDoS protection and scale.C
        3. In your hosted zone, build up an alias resource record. Similar to CNAME records, alias resource record sets can be configured at the zone apex.
        4. Check to see whether the redirect is functioning properly.

# TOOLS:

* 1. PHP (**for creating web applications**).
  2. EC2 instance on AWS cloud (**hosting Web App**).
  3. Distributed Load Testing on AWS / Monitoring metrics (**for load testing**).
  4. Hping3, HULK, Thor’s Hammer, LOIC, DDoSIM, HOIC and PyLoris (**for DDOS attack**), BYOB.
  5. Status check Monitoring metrics (**checks Status for your instances**).
  6. Elastic Load Balancing, Auto Scaling, Configuring Web ACLs and AWS Firewall Manager (**for mitigation DDOS attack**)

### **Creating a Web Application.**

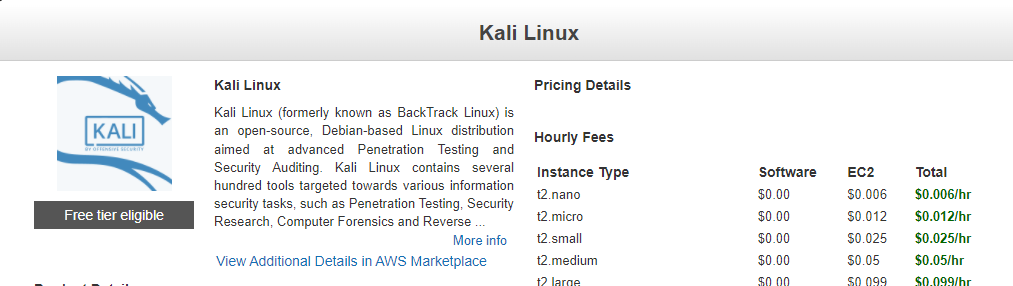
### **Data Flow Diagram**

### **Features**

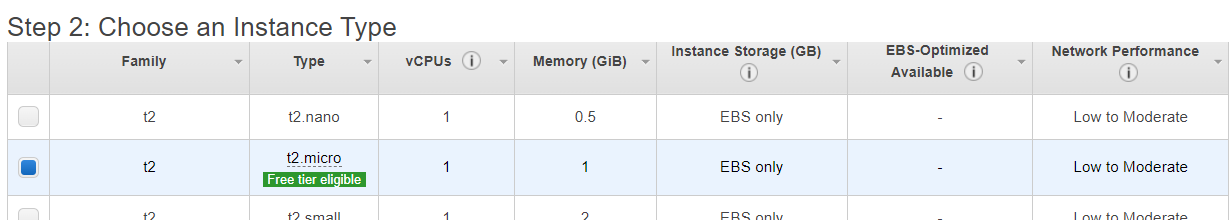
1. Customers have secure registration and profile management options.
2. Browsing the eMall to check what is available in each product category, such as apparel, kitchen accessories, bath accessories, food goods, and so on.
3. Creating a shopping cart that allows users to purchase a specified item before checking out with the whole cart.
4. Customers can email to shop with suggestions for things they'd like to see.
5. A safe method of checking out from the store (Credit card verification mechanism).
6. Customers are kept up to speed on the most recent items in the shop.
7. Uploading the Shop's most popular items in each category, such as apparel, kitchen accessories, bath accessories, food items, and so on.

### **Creating a Cloud Instance for the Web Application.**

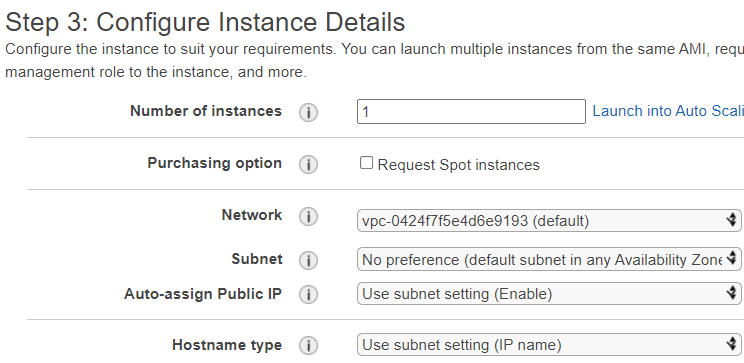
1. **Select the amazon machine image(AMI).**

The AMI is a software configuration template including operating system(OS), apps and server required for application, which is needed to launch our instance. We have the option of using the AMI offered by AWS, or community, or AWS Marketplace, or by creating own AMI.

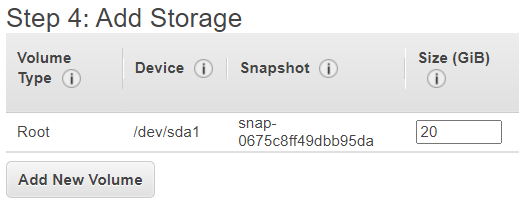
1. **Select type of the instance.**

Amazon’s EC2 offers a variety of instance’s types that are tailored to certain use cases. Instances are just like the virtual servers on which applications can be run. This come in a variety of CPU types, memory types, storage types, and networking configurations, giving us the freedom to select the best resource mixture as per our needs.

1. **Configuring the instance details.**

Configure the instances according to needs. We can use the same AMI to launch many instances, and assign an identity and access management(IAM) role to the instance.

1. **Add Storage.**

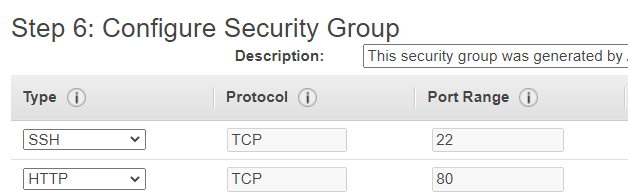
The storage device configurations listed below will be used to start our instance. We can expand our instance by adding new EBS volumes and instance store volumes, as well as changing the root (/) volume's specifications. Once an instance has been deployed, additional EBS volumes, but no instance store volumes, can be added.

1. **Adding a tag name.**

A tag name is a key-value pair that is case sensitive. Tag with the key Name and the value Web Server, for example, might be define volumes, instances, or both can have a duplicate of a tag applied to them. All instances and volumes will have tags applied.

1. **Configuring Security Group**

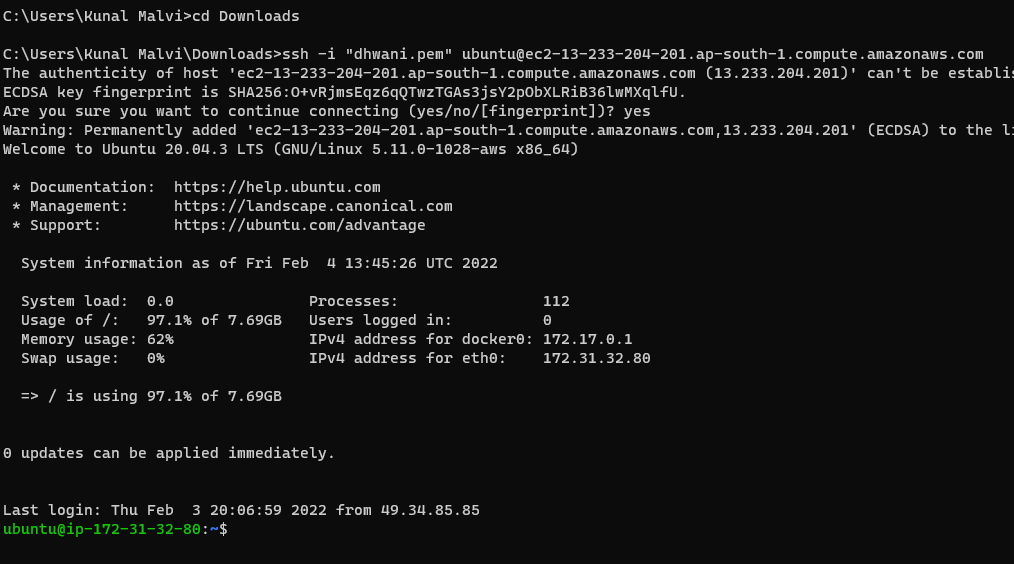
A security group is a set of (inbound outbound rules) firewall rules that control traffic entering the instance and exiting the instance. In this tab, we can create rules to enable specific types of traffic to access our instance. Let’s say we are going to host a web application then, we might need HTTP, HTTPS traffic to access the web application. And to host the application might need the access of the instance, so we may need SSH traffic.

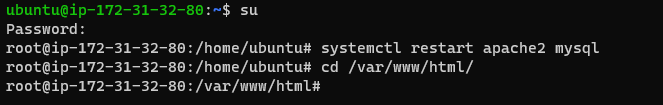


1. **Reviewing the details and Launch**

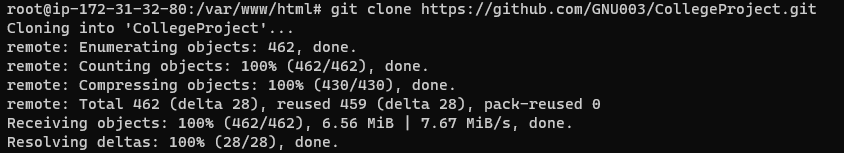
Once all the steps are followed, verify the details and launch the instance and it is ready to use.

### **Host the Web Application on Cloud Instance created.**

1. Firstly, connect to the EC2 instance created.
2. Switch to root user and restart the apache and mysql services.

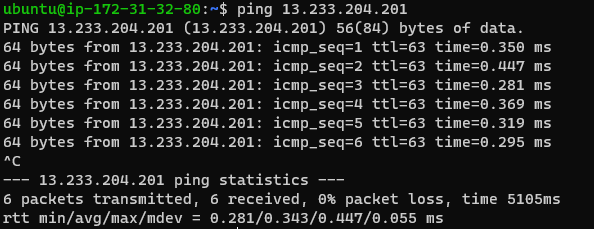


1. Clone the Web application from github.

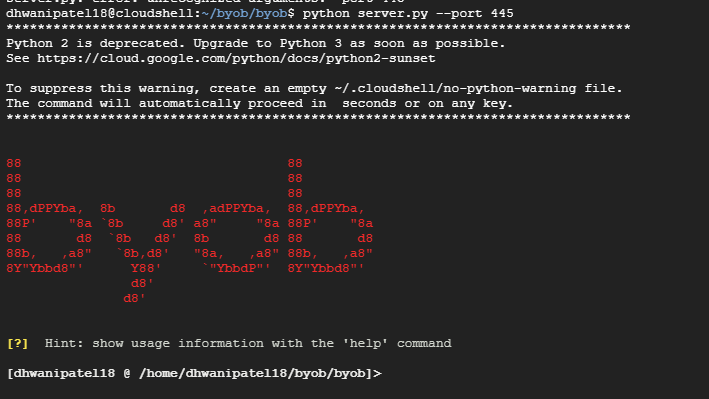
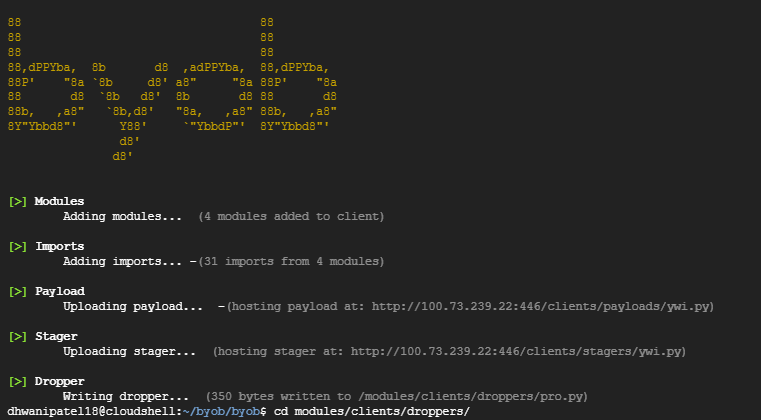
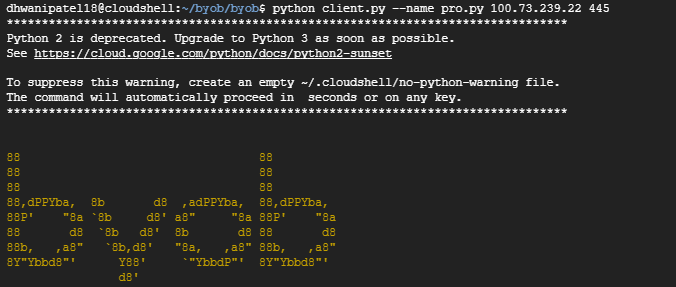
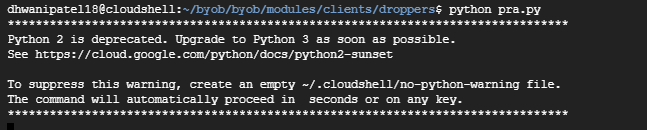
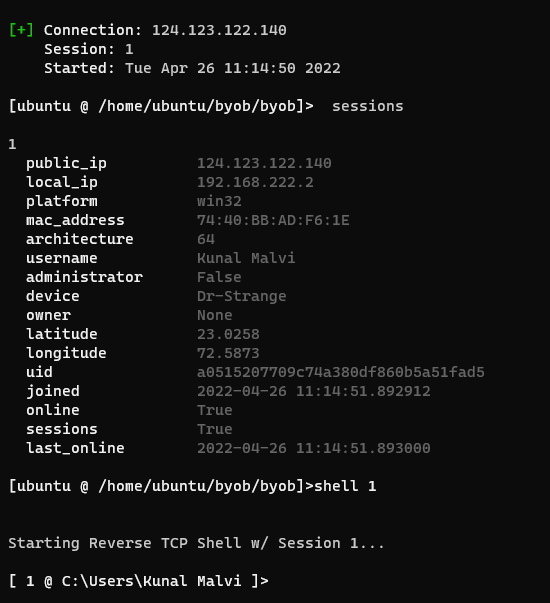


1. Check the website is hosted.

### **Performing DOS Attacks on the Cloud Computing Service (Website).**

1. Tried performing DoS using hping3 and Low Orbit Ion Cannon but no effect was observed.
2. So, we may need more than 1 system to bring the Cloud Computing Service down. In order to do so, we may need botnets to handle multiple servers.

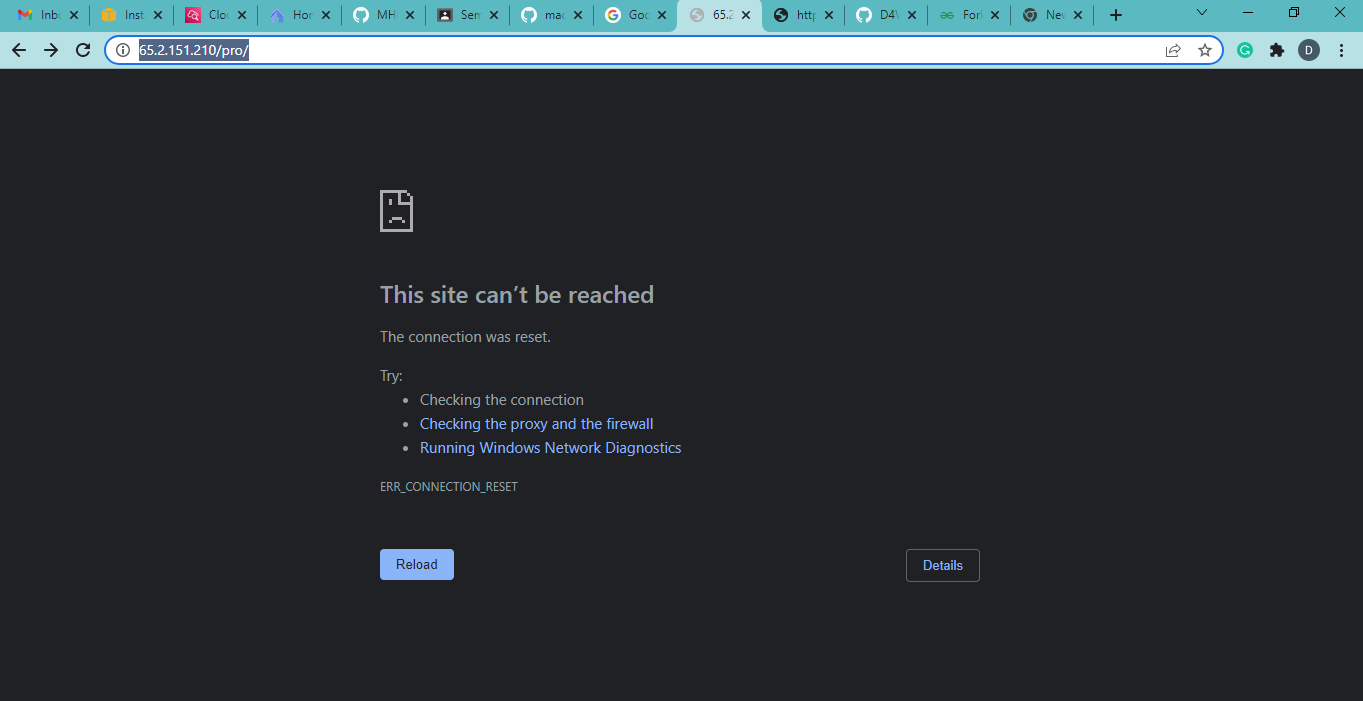
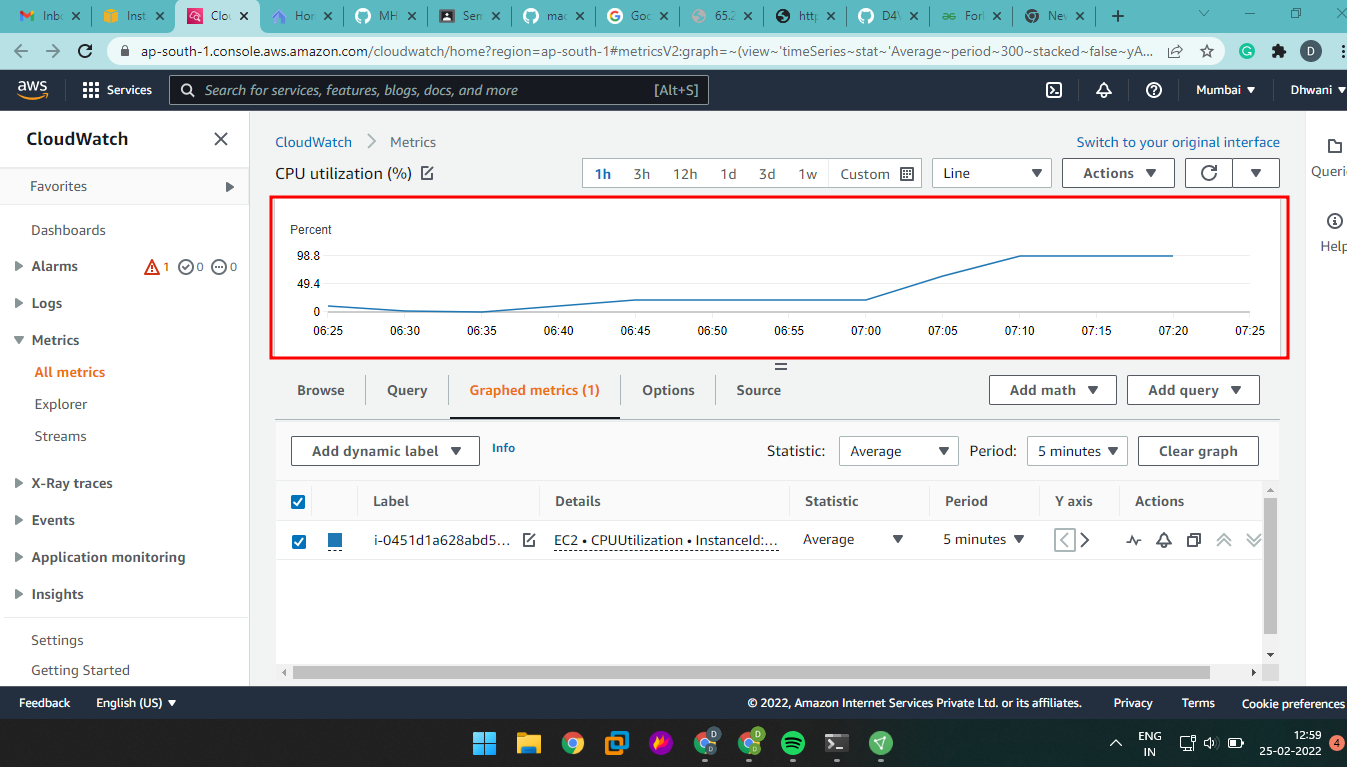
### **Setting Up Botnet server(ByoB).**

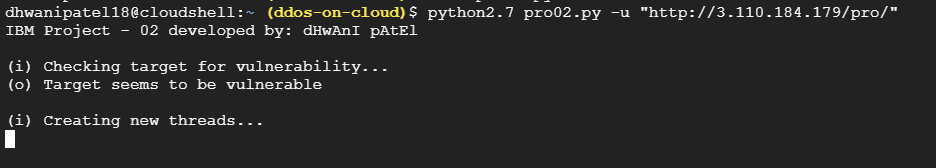
1. To host the botnet server, we created a new AWS instance.
2. BYOB (Bring Your Own Bot) is a botnet server and we will be using it to get reverse shells of systems, which we can use in the future to perform DDOS.
3. Attacker’s System:
4. The system used to generate the payload.
5. Now, the Attacker will be using Social engineering techniques to deliver the payload into the Victims system. After the successful execution of the payload, we may get the reverse shell of the client’s system.
6. Victim’s system.
7. Botnet server: 

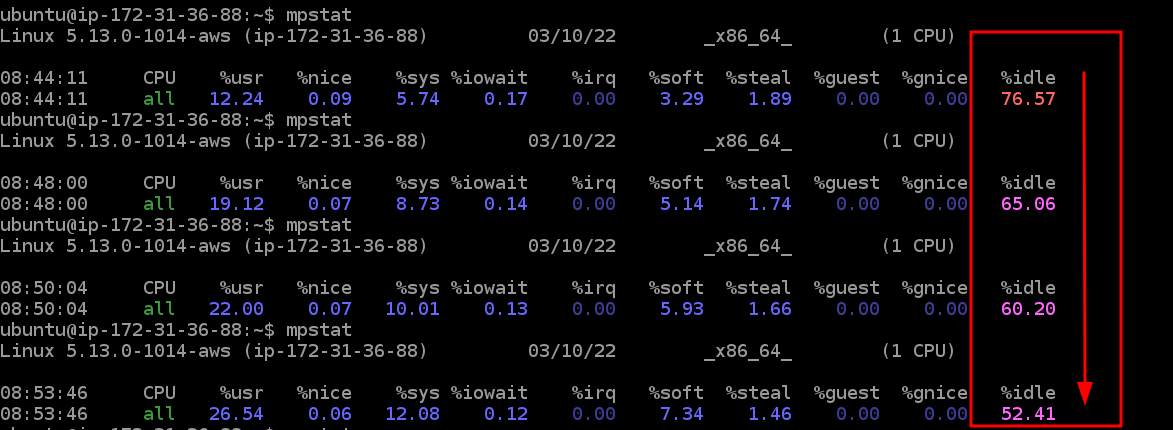
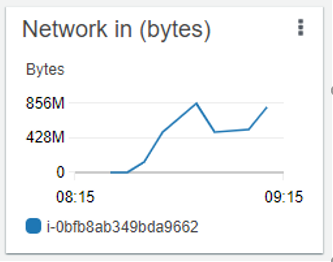
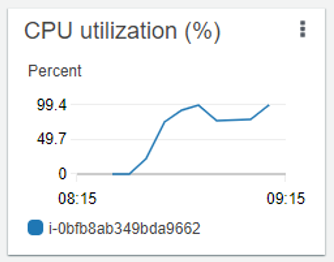
### **Performing DDOS Attacks on the Cloud Computing Service (Website).**

1. DDOS using RCE and Botnet Server:

|  |
| --- |
| **:(){ :|: & };:** |

Output:

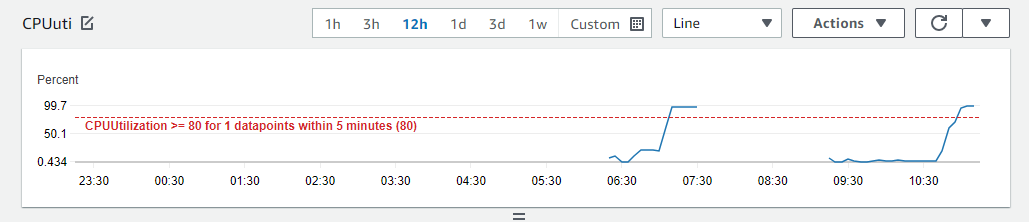
1. Sending range header to multiple overlapping ranges, we can increase the memory and CPU utilization. This is well known vulnerability in apache, which could be used to perform DDOS.
2. Shell 0: 
3. Running the script in **N** Shells

Output:  

1. Using multiple HTTP requests.
2. Http request:

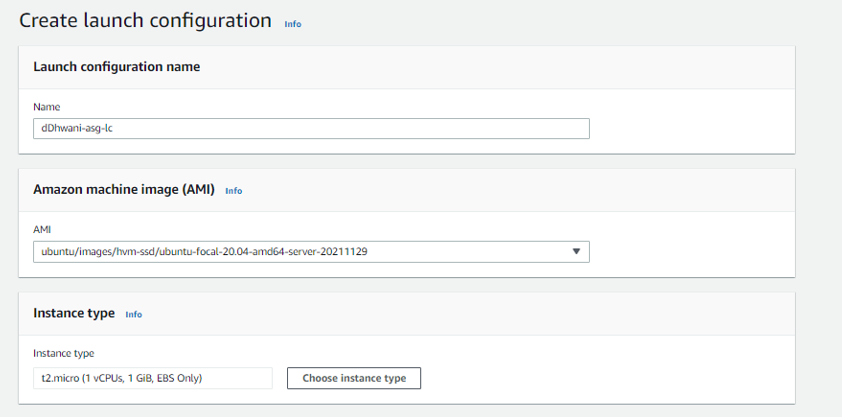
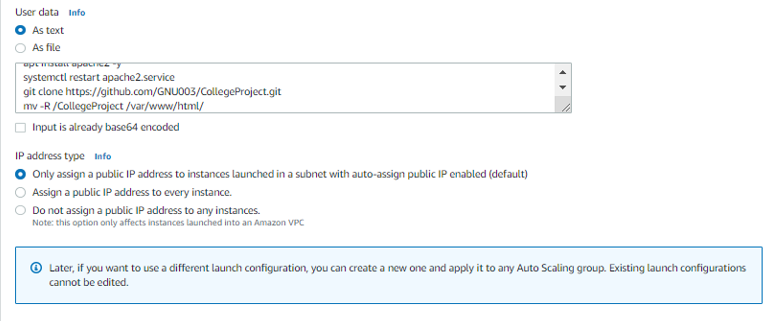
|  |
| --- |
|  |

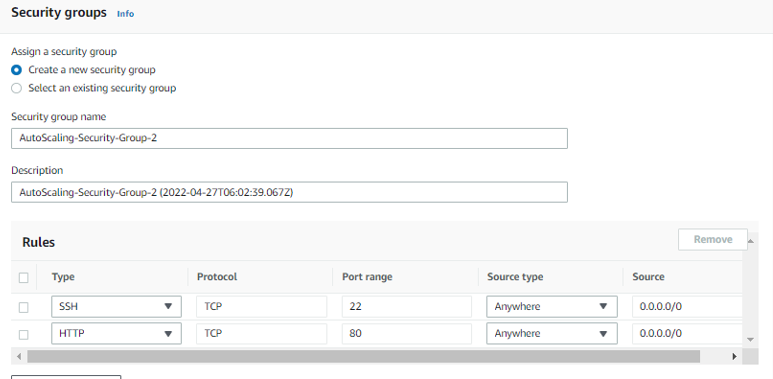
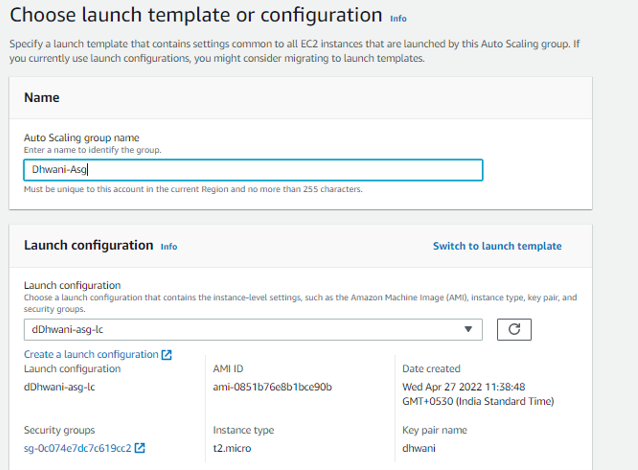
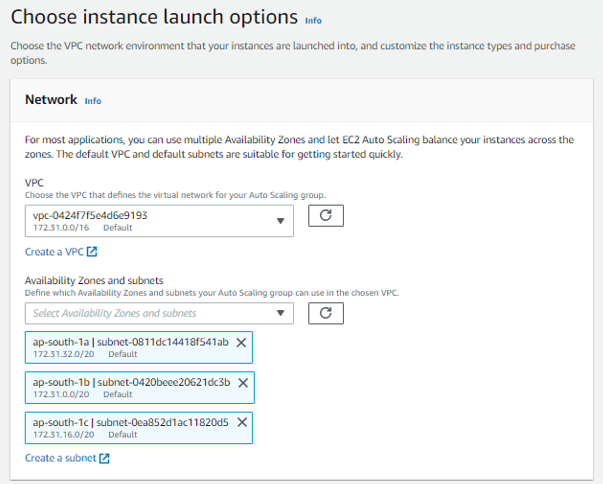
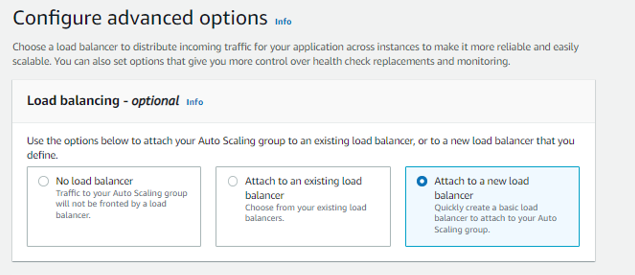
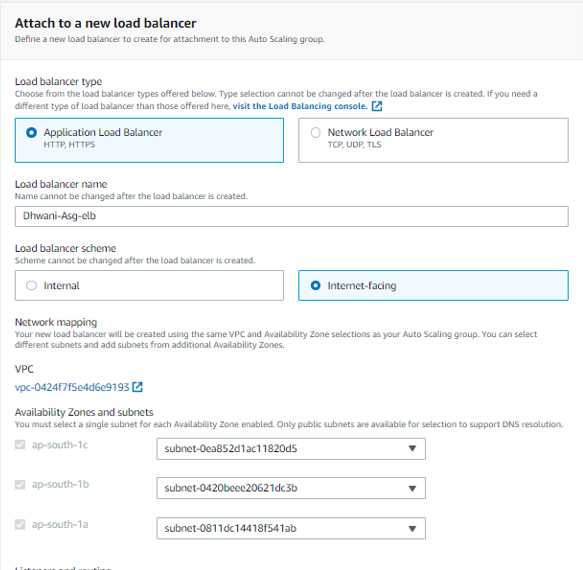
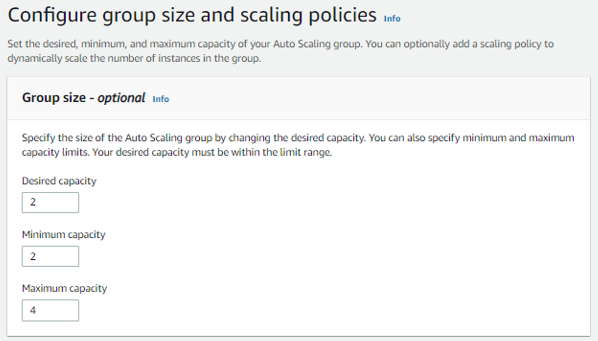
1. Collecting User Agents for sending multiple fake requests.
2. Running the script from **N** Shells.

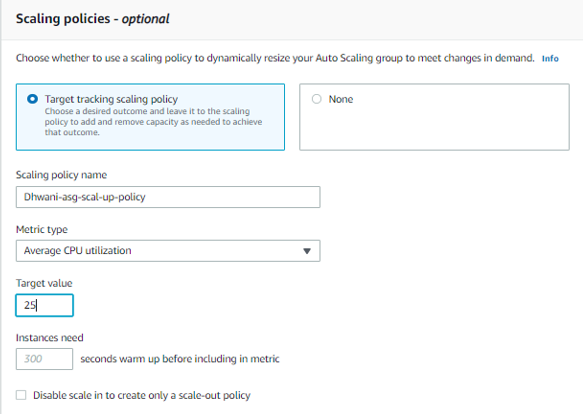
**Output:** 

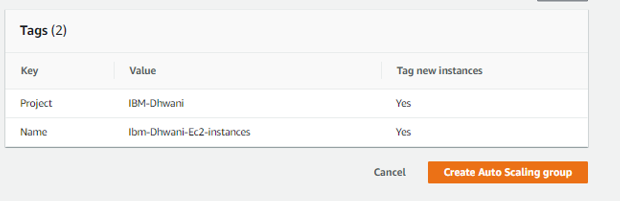
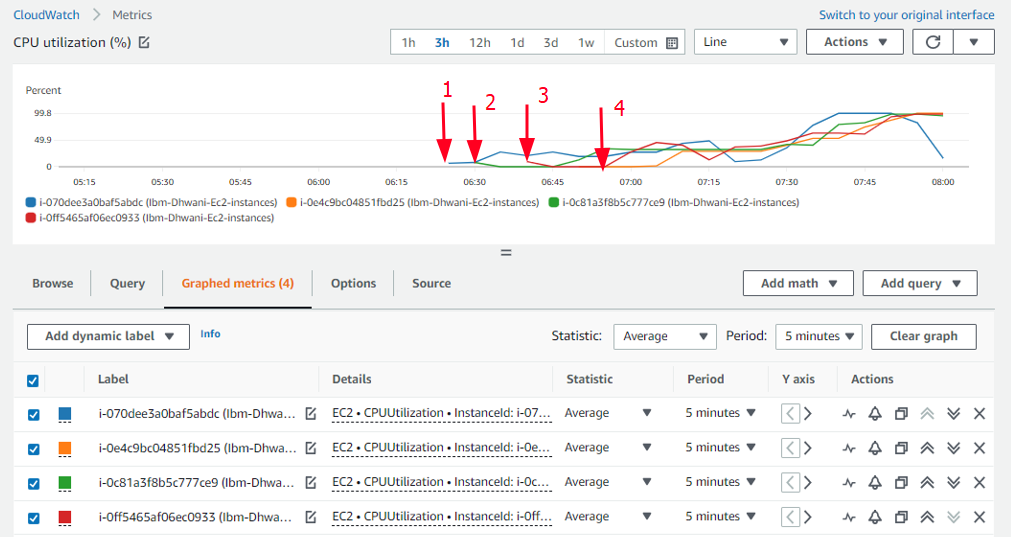
### **Mitigating the DDOS Attack**

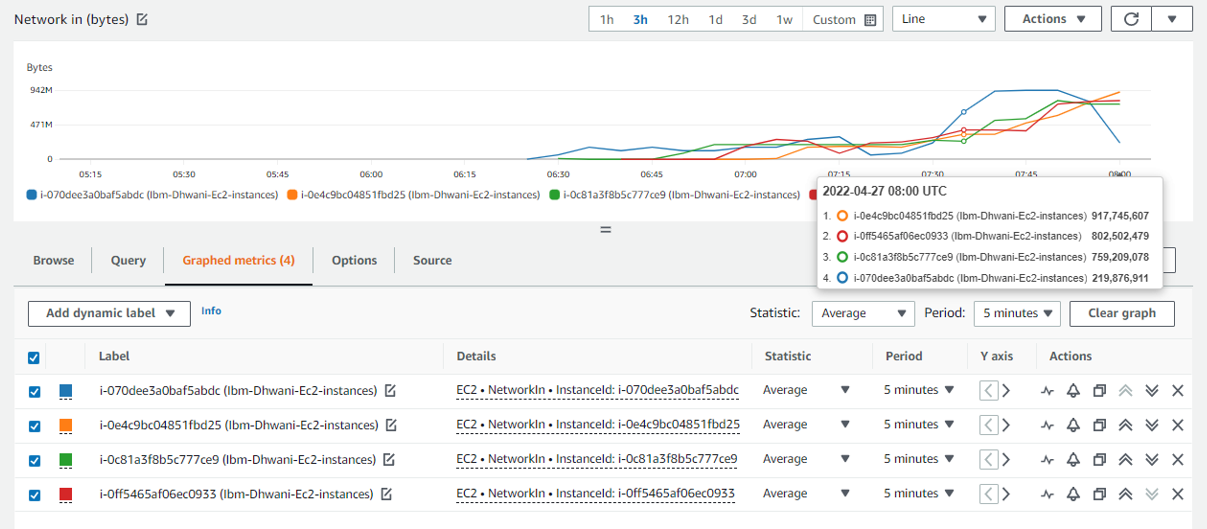
1. Use AWS Shield / AWS Shield Advanced.
2. Increase the bandwidth.
3. Enable AWS WAF.
4. Using Security Groups and network ACL.
5. Scaling instances / environment with load balancer.
   1. Create launch configuration.

* 1. Creating security group. 
  2. Launch configuration. 
  3. Choose instance launch options. 
  4. Attaching a new load balancer.  
  5. Configure group size and scaling policies. 



* 1. Reviewing and launch. 
  2. Checking the Auto-scaling by performing DDOS. 



**Conclusion**

DDoS attack is one of the major threat right now, and this works against the availability of cloud service. We demonstrate that defence techniques to protect against DDoS assaults are not always successful on their own by launching DDoS attacks against Cloud Computing services. The necessity of the hour is to combine multiple mechanisms to create hybrid defence mechanisms.

### **Future work**

Till now we have completed Hosting a service on AWS EC2 instance and a client-server botnet. Furthermore, using the botnet server and the reverse shell access of multiple computers, we tried performing some of the DDoS attacks.

Here we have performed DDoS in application layer, we can try bypassing the default Firewall of AWS and could perform DDoS on multiple layers. We have implemented auto-scaling as mitigation of DDoS. AWS Shield, Security groups with ACL, Enabling service log for deeper analysis, enabling AWS WAF are some of the other paid mitigation, which will help the cloud user improve the user experience of the cloud computing services.

**Initial project reconnaissance and planning**

### **Creating a Web Application.**

### **Creating a Cloud Instance for the Web Application.**

### **Host the Web Application on the Cloud Instance created.**

### **Performing DOS Attacks on the Cloud Computing Service (Website).**

### **Setting Up Botnet server(ByoB).**

### **Performing DDOS Attacks on the Cloud Computing Service (Website).**

**Mitigating the DDOS Attack**