|  |  |
| --- | --- |
| Name: | Sandeep Chowdary |
| Lab User ID: | 23SEK3324\_U03 |
| Date: | 09-01-2024 |
| Application Name: | OWASP Juice Shop |

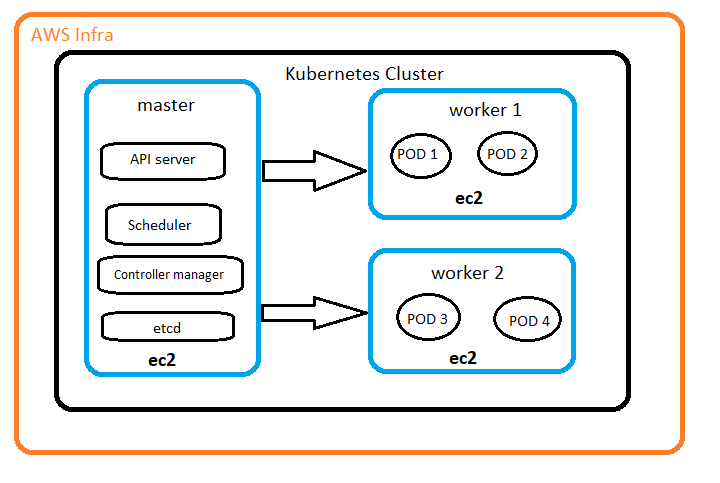
**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)



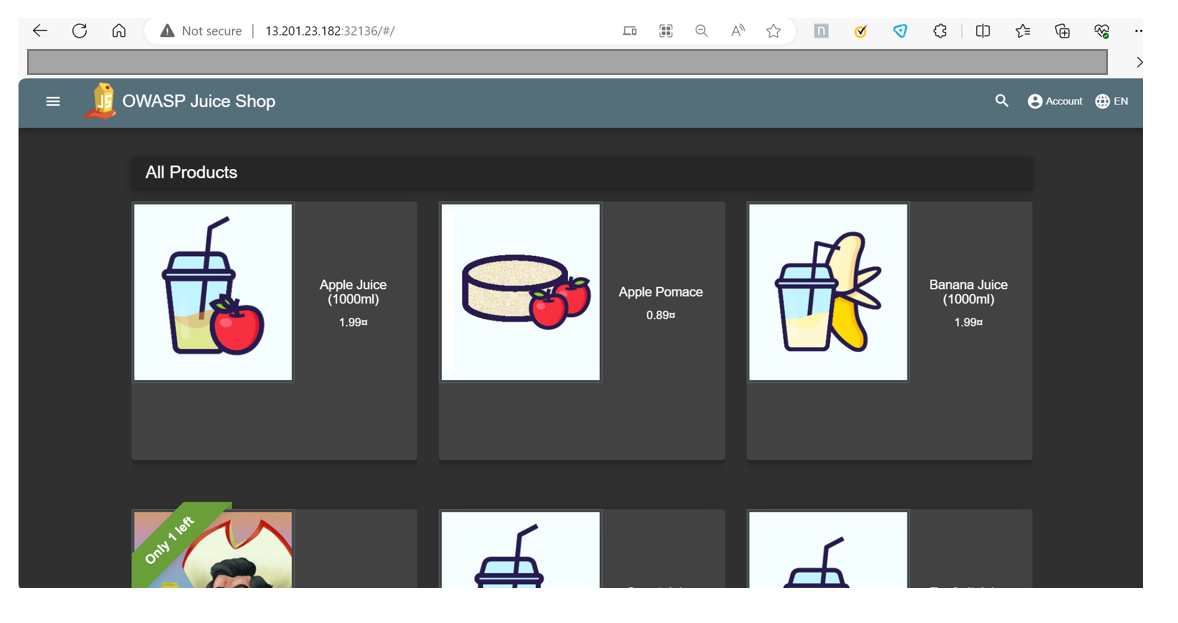
Kubernetes Service

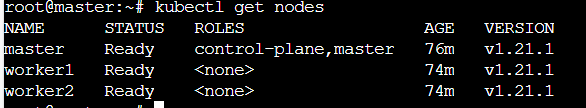
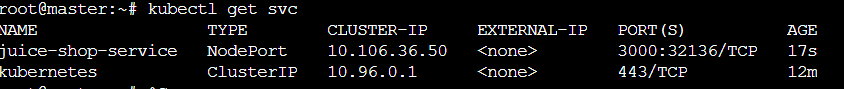
http://3.110.46.1:32136/

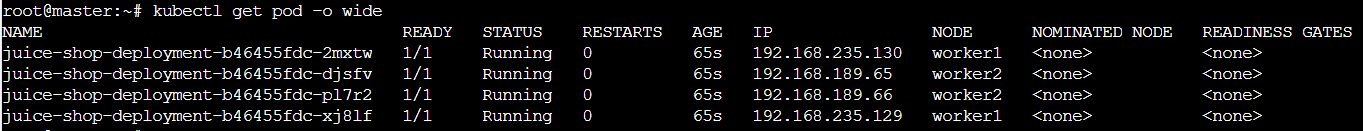
http://13.201.23.182:32136/

Web browser

**USERS**





Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

The juice shop application user interface is user friendly. Its has different components in the application where a user

a user can easily navigate. User can Users can browse products, add items to their shopping cart, and proceed with

the checkout process. The overall UI is good and responding smoothly with the user clicks.

The application is running in Kubernetes cluster as a deployment with 4 replicas of the application.

The system contains two virtual machines, and working properly. The application is running on port 32136.

The server is properly responding to the requests. The response time is good with no errors.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**Known**

Even when a server is suddenly stopped the application is accessible.

The system's resource limits, especially in terms of CPU, memory, and storage, are well-understood under normal conditions, but potential issues may arise when these limits are exceeded unexpectedly.

**Unknown**

**Unknown**

**Known**

Scaling mechanisms are assumed to handle increased load, but there may be specific scaling challenges or limitations in certain scenarios that have not been identified

We don’t know what will happen if we Shut down the entire Kubernetes cluster

1. Even when a server is suddenly stopped the application is accessible.

2. The system's resource limits, especially in terms of CPU, memory, and storage, are well-understood under normal conditions,

but potential issues may arise when these limits are exceeded unexpectedly.

3. Scaling mechanisms are assumed to handle increased load, but there may be specific scaling challenges or limitations in certain scenarios that have not been identified

4. We don’t know what will happen if we Shut down the entire Kubernetes cluster

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

1)About the application:

OWASP Juice Shop is an intentionally insecure web application designed for security training, awareness

demos, and Capture The Flag (CTF) competitions. It is developed and maintained by the Open Web

Application Security Project (OWASP), an nternational non-profit organization focused on improving

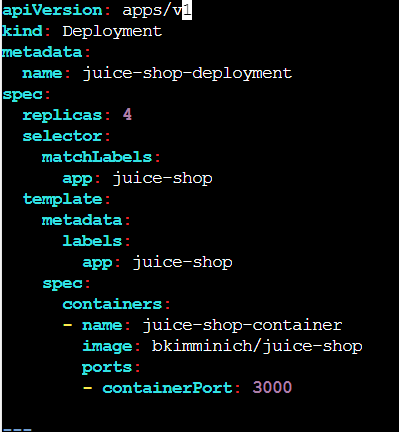
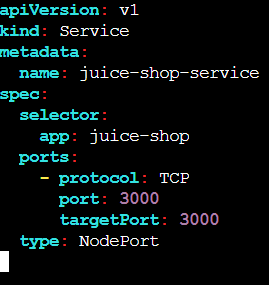
the security of software.

2) Making the application live:

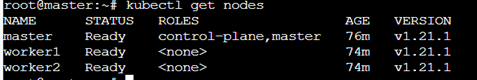
Created a Kubernetes cluster of 3 ec2 instances.

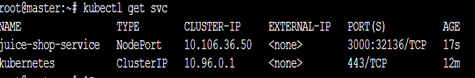
Pulled the application image from the dockerhub.

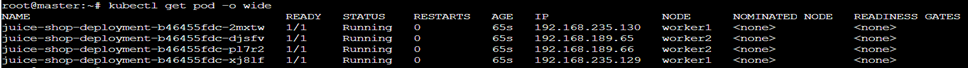
Create a deployment and a service using the yaml file:

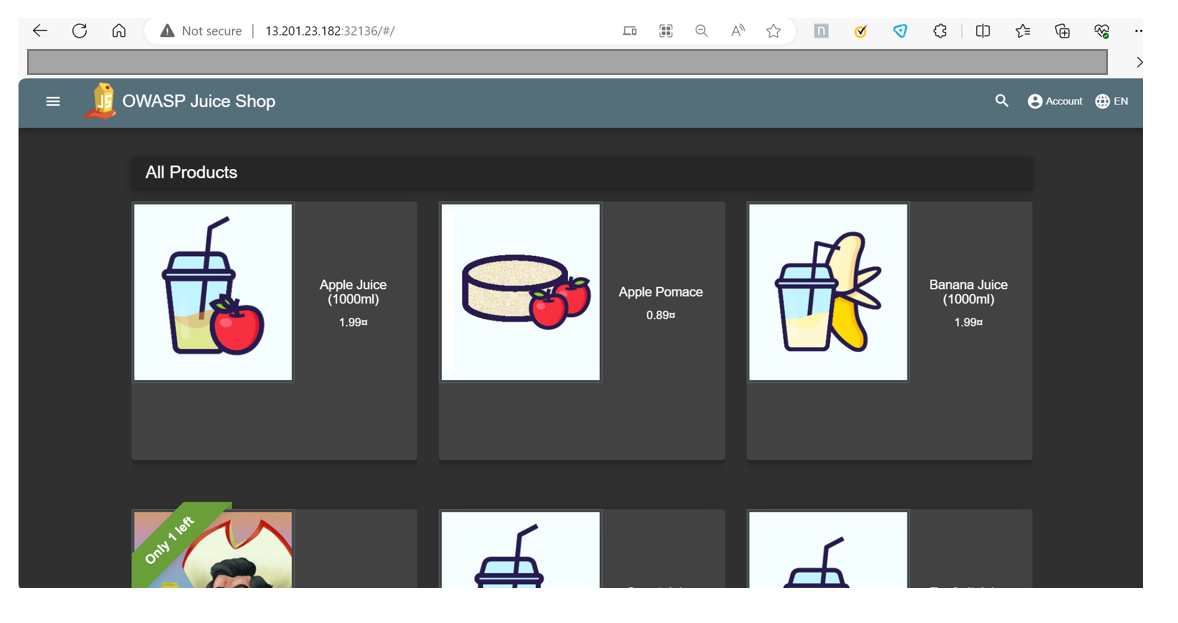
3) Below are the Kubernetes nodes, services, and pods that are running





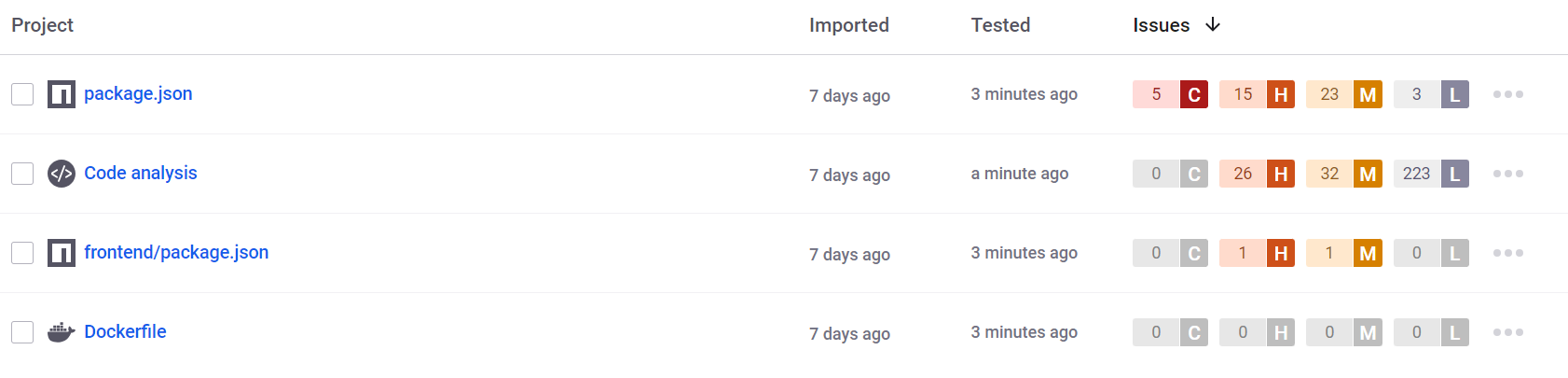


4) Checking whether the application is live:

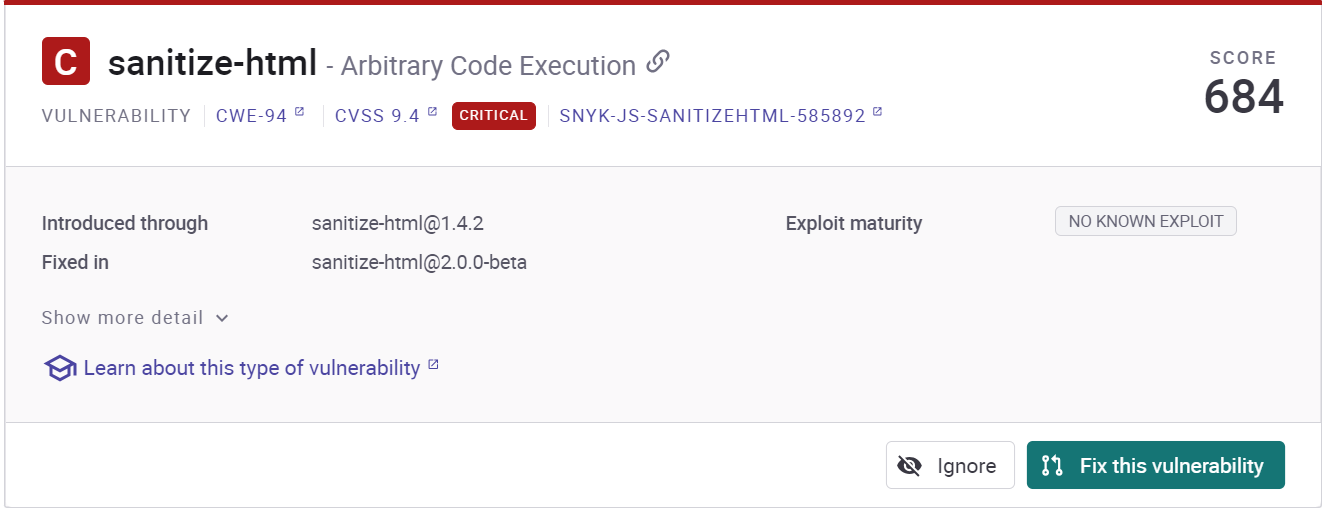


5) now scanning the vulnerabilities of the web application using the snyk tool.

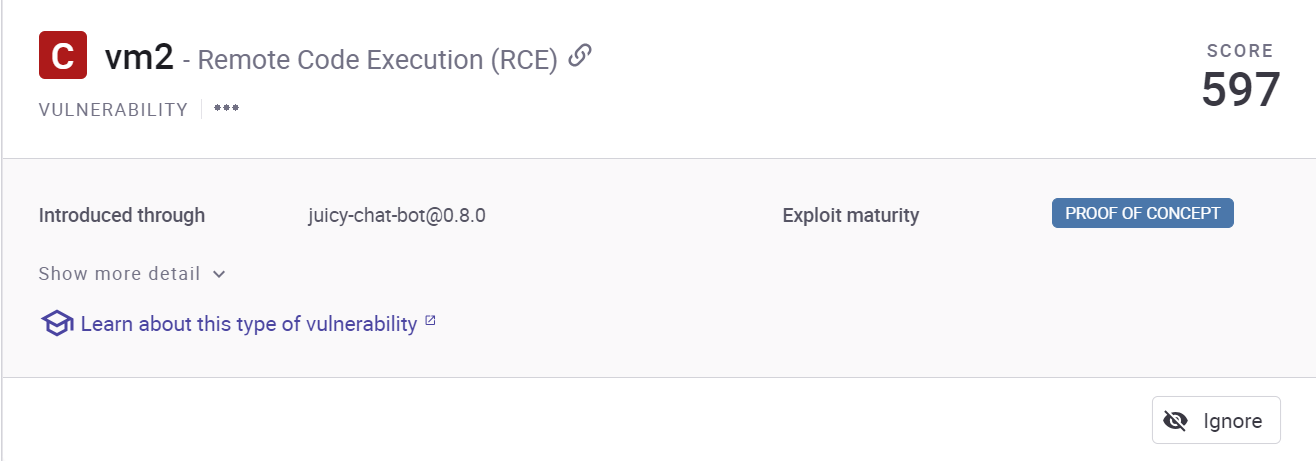
6) In the snyk I imported the git repo and snyk scanned the git repo for the vulnerability:



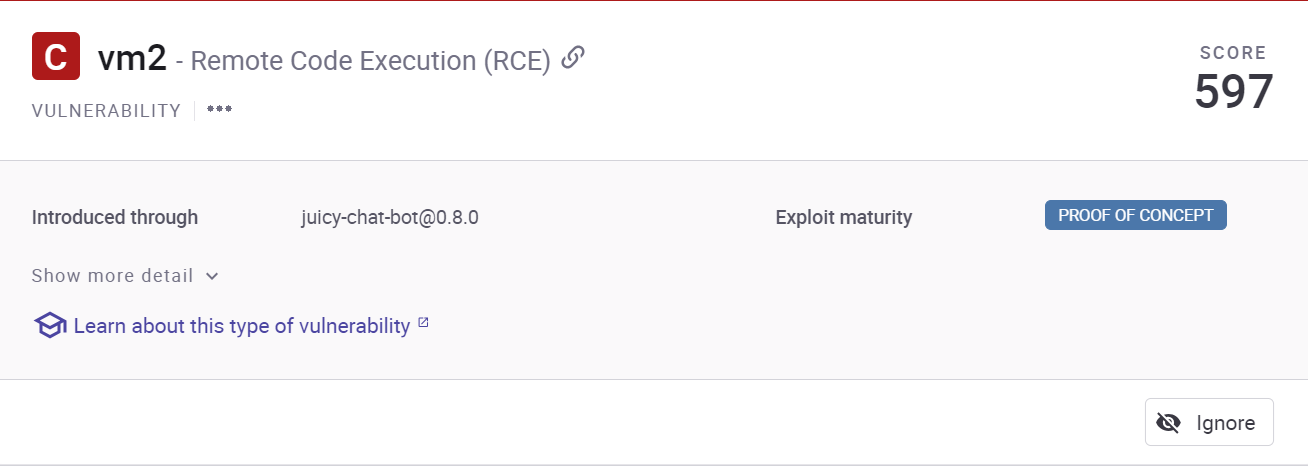
7) **package.json is dependency that has total of 46 vulnerabilities: Critcal : 5**

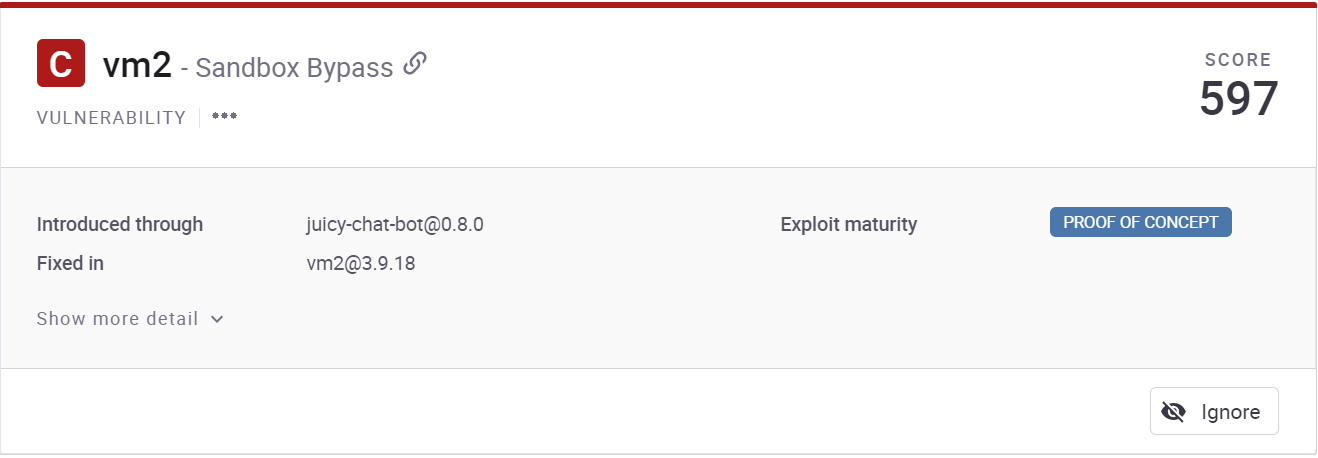
7.1.

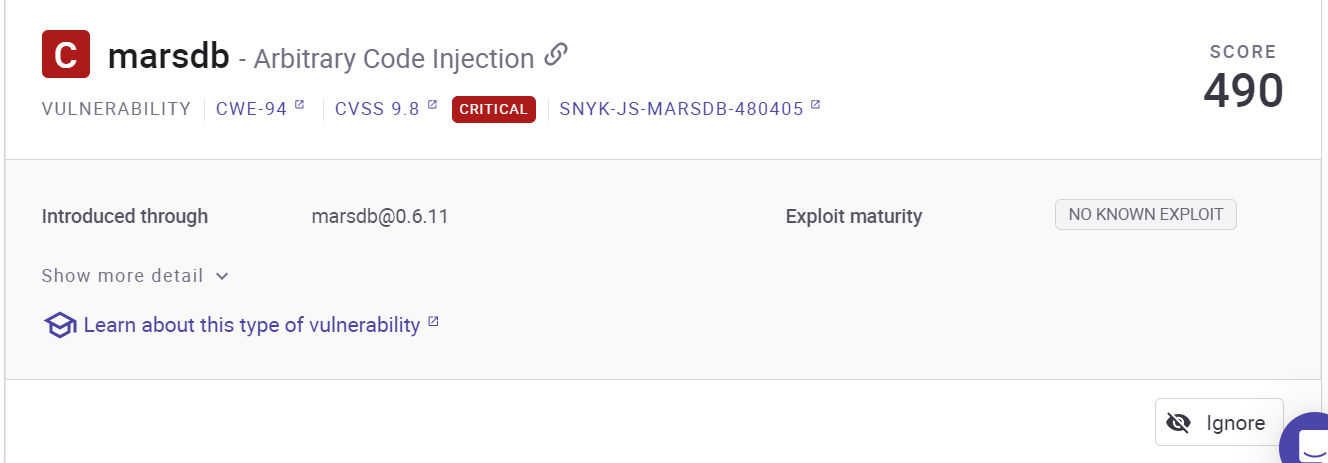
7.2.



7.3.



7.4. 

7.5. 

8)**vulnerability scanning by zap:**

8.1.Cross-Domain Javascript source file inclusion : The page includes one or more script files from a third-party domain.

Solution : is Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

8.2. Information Disclousre suspicious comments : The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.

Solution : Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

8.3. Conent security policy : Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attack.

solution : Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

8.4. Permission policy header not set : Permissions Policy Header is an added layer of security that helps to restrict from unauthorized access or usage of browser/client features by web resources. This policy ensures the user privacy by limiting or specifying the features of

the browsers can be used by the web resources.

Solution : Ensure that your web server, application server, load balancer, etc. is configured to set the Permissions-Policy header.

8.5. Dangerous Js function : A dangerous JS function seems to be in use that would leave the sitevulnerable.

Solution :See the references for security advice on the use of these functions

9)**Performing Chaos engineering by using Gremlin:**

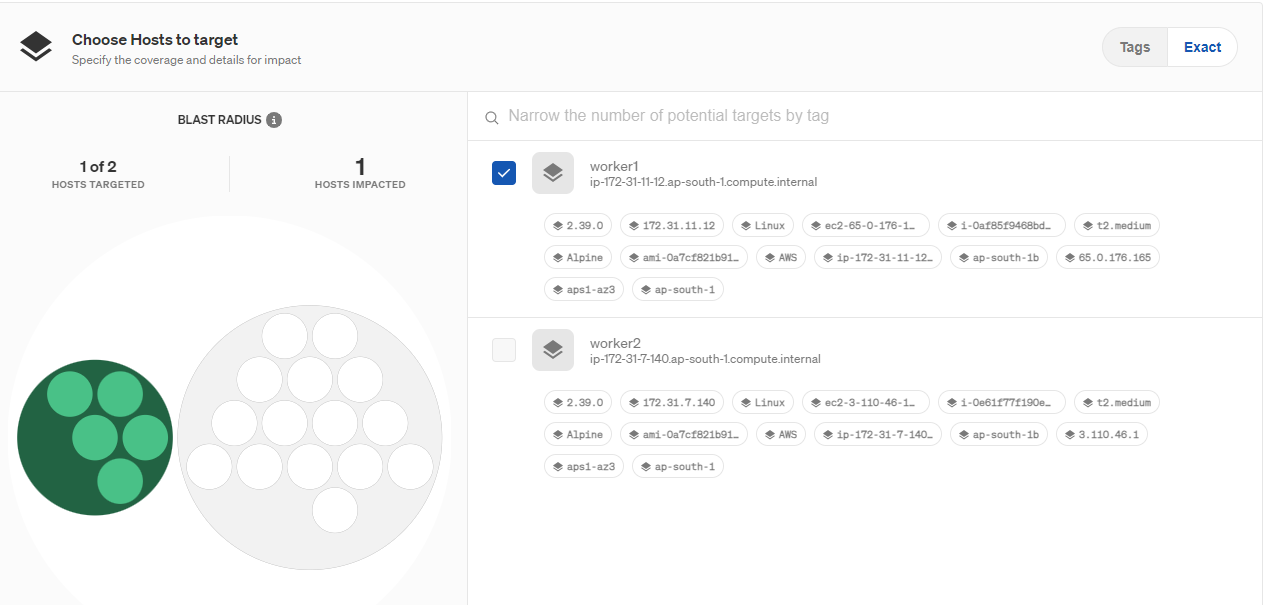
With the help of gremlin UI based web application I have performed a simple experiment to check the behavior of

the system in a disruptive environment.

For this I logged in to my gremlin account.

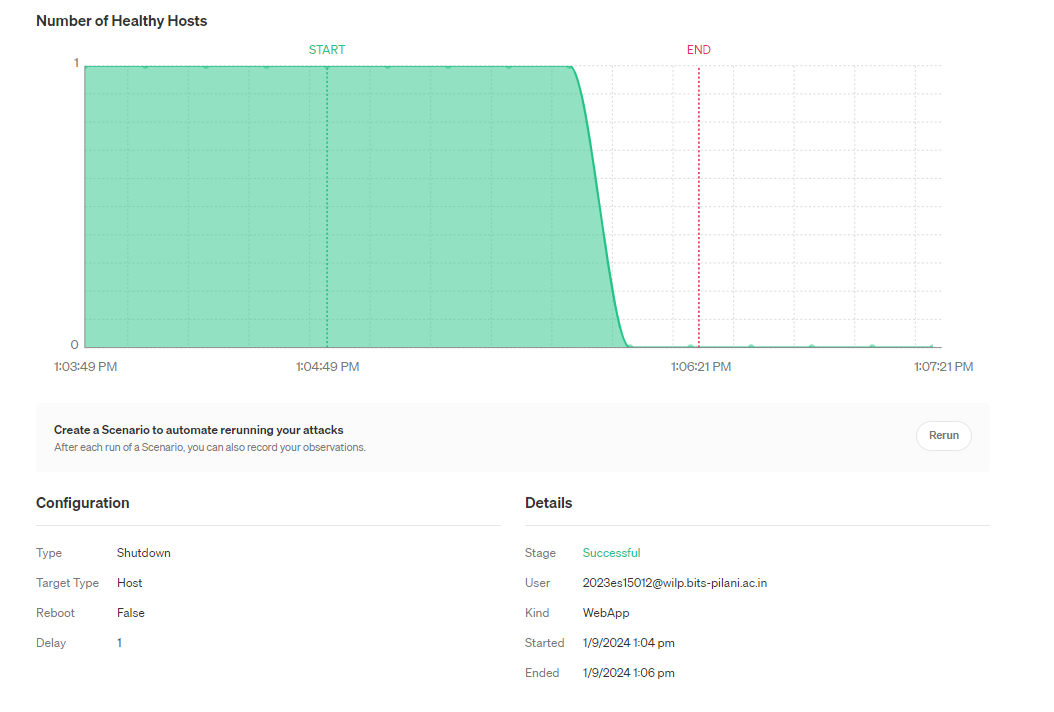
I installed the gremlin agent (helm) over the master node in the cluster.

In the gremlin the nodes are attached.

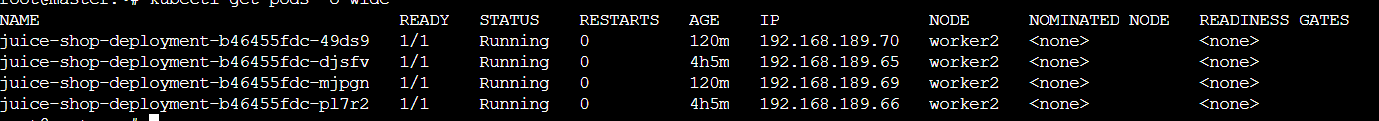


Then performed an experiment where I targeted the system state of the worker1 node.

The worker node is shut down.



After the worker1 node is shut down the Kubernetes allotted all the pods to the workder2 node.



Now I am not able to access the website with workernode1 ip address.

Conclusion :

The Kubernetes cluster is working properly. When one instance is shut down then all the replicas are still running and

It is running on worker2 node. But the site is not reachable because we are using the worker1 public ip. So for this

The solution would be using the elastic ip.