|  |  |
| --- | --- |
| Name: | Mazhar Hussain C K |
| Lab User ID: | 23SEK3324\_U06 |
| Date: | 09-01-2024 |
| Application Name: | 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)

Deployed in Node 2

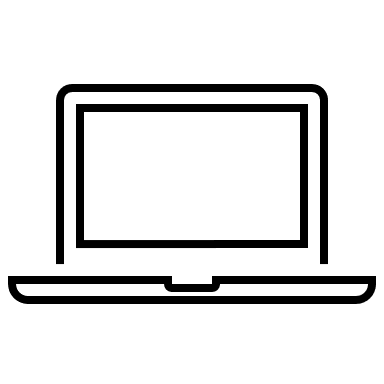
EC2 **Instances**

Kubernetes

Cluster

Node-2

Node-1



Accessing the web app through Ip address

We have used AWS EC2 for the deployment of OWASP Juice Shop. We Created three ubuntu EC2 deployments and allowed all traffic to come for the testing process. Created a Kubernetes Cluster with one master node and two other worker nodes.

One YAML file is used for the deployment of Juice-Shop as deployment, upon successfully creating the deployment, another yaml file is used for defining service to access the web app through internet.

The web app can be accessed using the nodes Ip and port number.

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)

Performance: The application is able to respond quickly for user interactions under normal conditions, without any

latency issues, even though it was designed as intentionally vulnerable app.

Resiliency: As it was deployed as deployment is Kubernetes, any failure in the node in which it is was hosted will

result in the instant deployment of the app in another node.

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 e.g.: "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.")



We are aware if there is any failure in the node which the web app has been deployed it will immediately get redeployed to the next working node available, in our case it is the remaining one node.

**Known**

Potential impact on application performance and user experience during Kubernetes auto-scaling.

**Unknown**

**Unknown**

**Known**

Unanticipated effects of chaos experiments on system behavior.

Impact of potential external factors on our chaos experiment, such as network outages

Unexpected errors resulting from simulated chaos conditions.

As we are about to conduct our chaos experiment on the OWASP Juice Shop deployed on our Kubernetes cluster, we

hypothesize the following outcomes:

Pod Failure: In the event of any pod or the node itself in our Kubernetes cluster failing, the system will automatically

spin up a new pod to maintain the operational stability and effectiveness of the application. Consequently, we expect

minimal downtime- if any - during this process.

Along with that this is an intentionally vulnerable app, so we expect to find a good number of vulnerabilities.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis)

**Preparation**

Before Conducting Chaos experiments, we need to do vulnerability scanning for vulnerability analysis. The tool that

have been used here for scanning is snyk. This tool searches for vulnerability in the source code repository, in our

case its GitHub.

For Conducting the chaos experiments we have used Gremlin.

Gremlin Agent is installed on the Kubernetes cluster through helm method. We are going to target the host in which

Juice Shop Deployment is deployed and force it to go on a shutdown to check the resiliency.

Before conducting the experiments, we made sure that all other nodes are in running state.

**Implementation**

The GitHub repo has been imported to the trivy scanner for vulnerability scanning.

It has analyzed the package.json file along with the source code, frontend and the dockerfile.

Upon successful completion of adding gremlin agent to Kubernetes cluster, we are going to conduct a shutdown

experiment on the Worker node in which the app is live without a reboot option with a delay of 2 minutes so that

we can again make sure that all the backup process are in place.

**Observation and Analysis**

Vulnerability Assessment:

There are 329 vulnerabilities in the juice shop repository. Among them 5 were critical 42 high vulnerabilities,56

medium and 226 low vulnerabilities. The critical vulnerabilities were inside the package.json,some of them are

Remote code execution, Arbitrary code execution and arbitrary code injection.

along with that other 15 high,23 medium and 3 low vulnerabilities was also found.

There was a total of 15 vulnerabilities which don’t have any supported fix.

In the code analysis 26 high,32 medium and 223 Low vulnerabilities were found. Some of the high vulnerabilities

were Cross-site Scripting, multiple instances of path traversal, NoSQL injection, Hardcoded Secrets and so on.

Chaos-Experiment:

Chaos Experiment is done on the host using gremlin in which a shutdown process was initiated which resulted in the

Shutdown of the host and creation of the web app deployment in the second node, the time taken for redeploy was

not more than 3 seconds. The cluster was accessed through node port without load balancer because of that we had

to use the Ip of the second host in order to access the webpage, which is not very efficient.

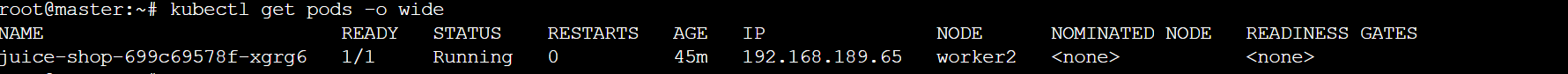
We also run a second shutdown process after starting the stopped node, this time even though it was deployed back

to the first node, we couldn’t access it, this was because of the dynamic Ip address in aws, in which the Ip of a

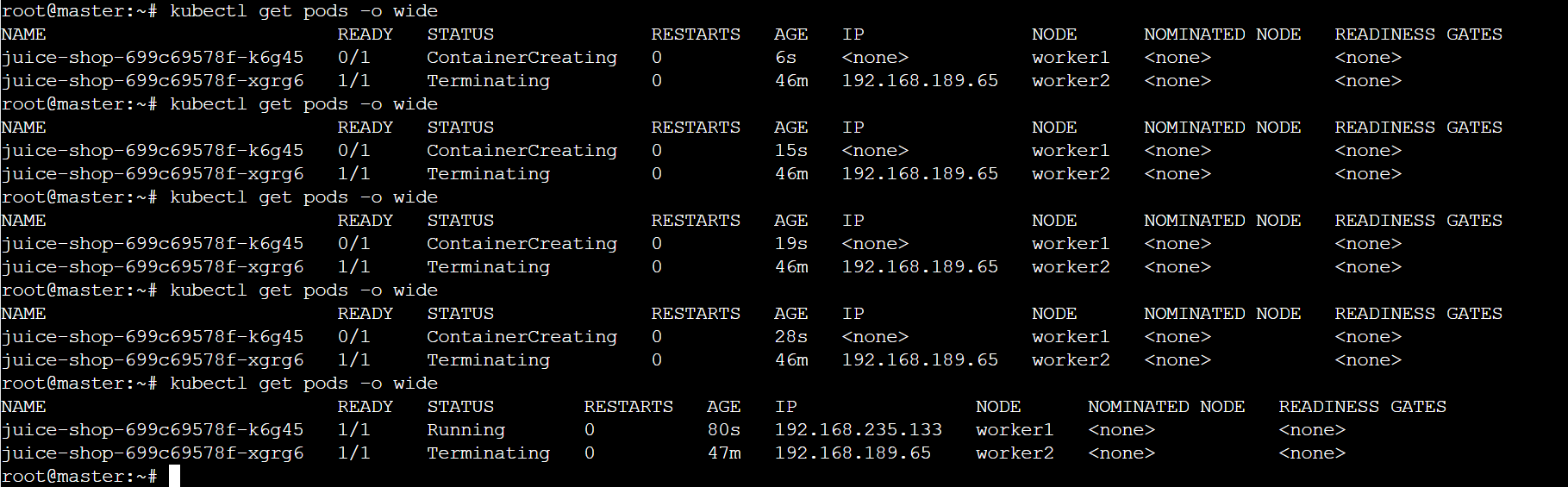
machine changes after its gone through a shutdown-start process. To overcome the issue, we had to add static Ip

address to the nodes in Kubernetes cluster

Before Chaos Experiment



During Chaos Experiment



As you can see how the container is deployed on the other node when the chaos experiment is done on the first

node