|  |  |
| --- | --- |
| Name: | Rishitha Pudi |
| Lab User ID: | 23SEK3324\_U01 |
| Date: | 09-01-2024 |
| Application Name: | Data-Dog-Java-application |

**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)

EC2

Docker

Registry

Docker Host

Client

container

Docker Build

Image

Docker

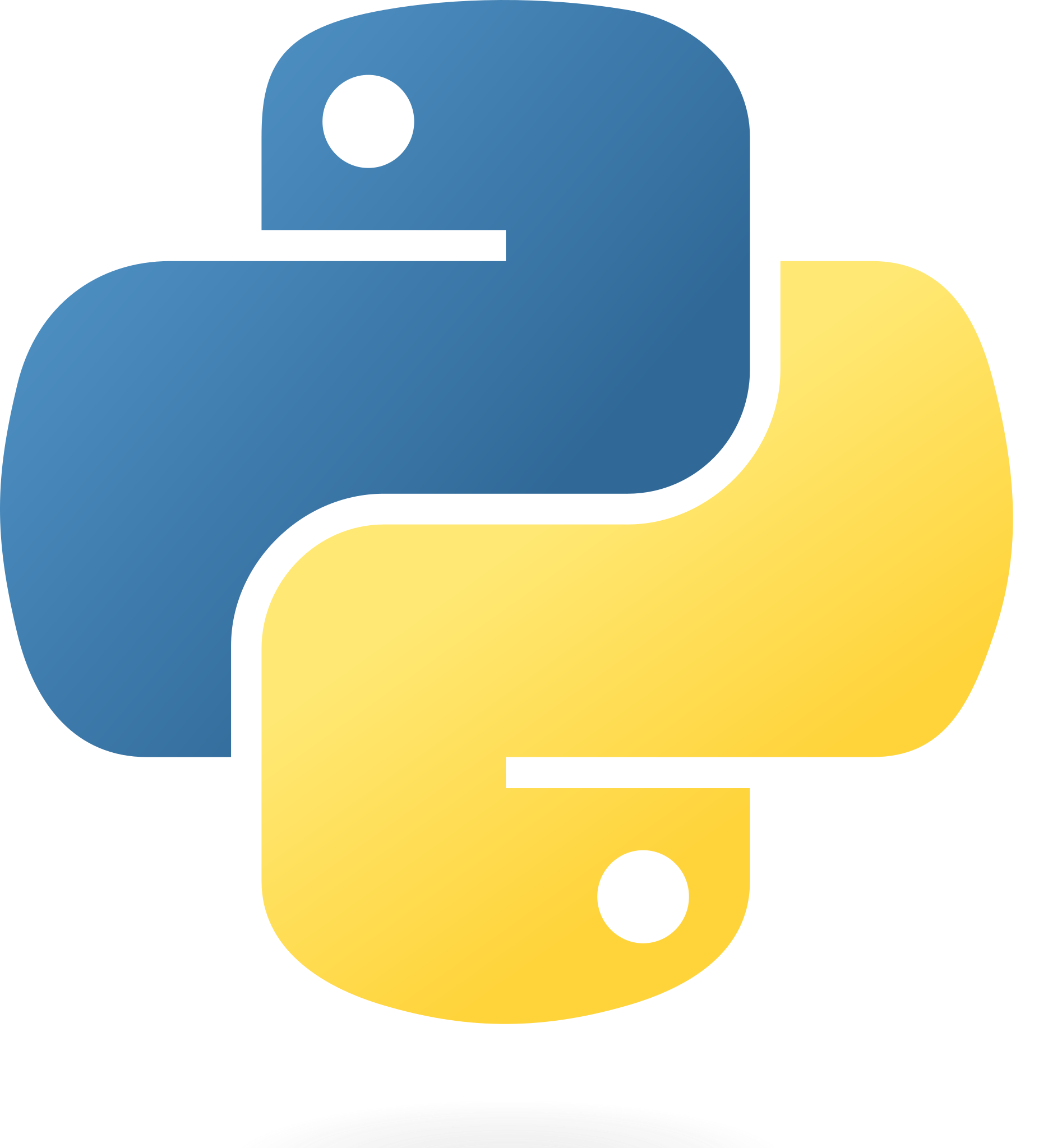
Daemon



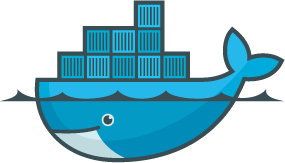


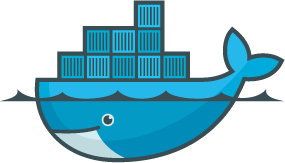


Images







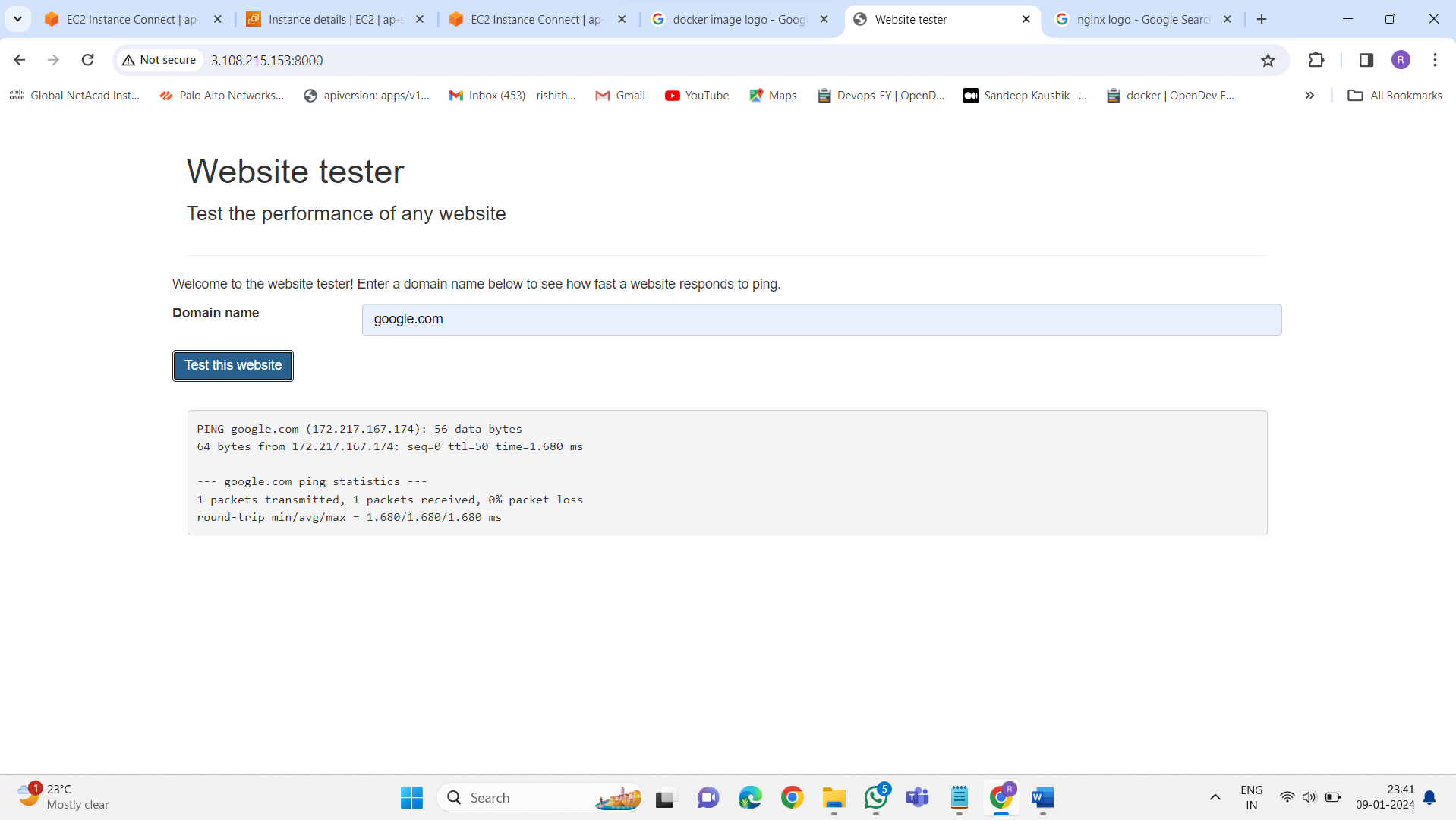


Docker Pull

Docker pull

3.108.215.153

http://3.108.215.153:8000



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 web server starts, listening on defined ports like ip address: 8000.

A user accesses a website hosted on this server via a web browser.

The website is divided into 3 parts. header division, domain search division and the ping result.

If the user gives the correct domain name, then the website start pinging that domain name.

At the ping in division, it will show all the results getting from that website

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.")



Chaos engineering hypothesis scenarios

Engineers test the system's capacity by gradually increasing user load until it reaches and surpasses the expected operational limits. While they know overloading the system might cause issues, the specific breaking point or the precise failure mode might be unknown

Engineers intentionally shut down a database server during non-peak hours to observe how the system reacts. They expect service degradation or failures in specific functionalities that rely on the database

**known**

Through load testing, engineers discover that a particular microservice experiences significant latency only when multiple users concurrently access a specific feature.

Engineers randomly throttle network bandwidth or introduce intermittent latency into different parts of the system to simulate unpredictable real-world conditions.

**unknown**

**known**

**unknown**

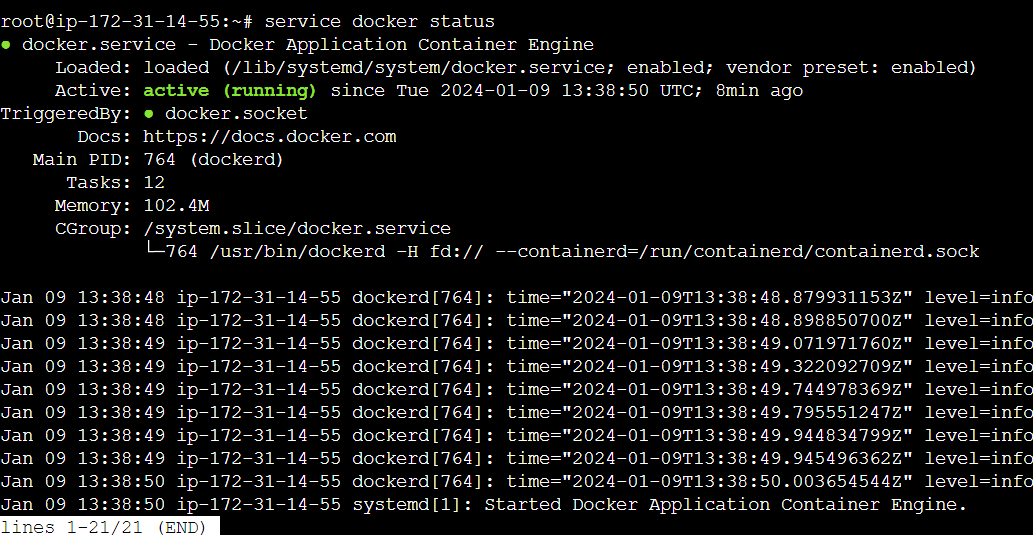
Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Overview**: [vulnerable-java-application](https://github.com/DataDog/vulnerable-java-application) is application created by the data dog. This application is purposely vulnerable and can trivially be hacked. Using this application we can perform some vulneerabile scanning.

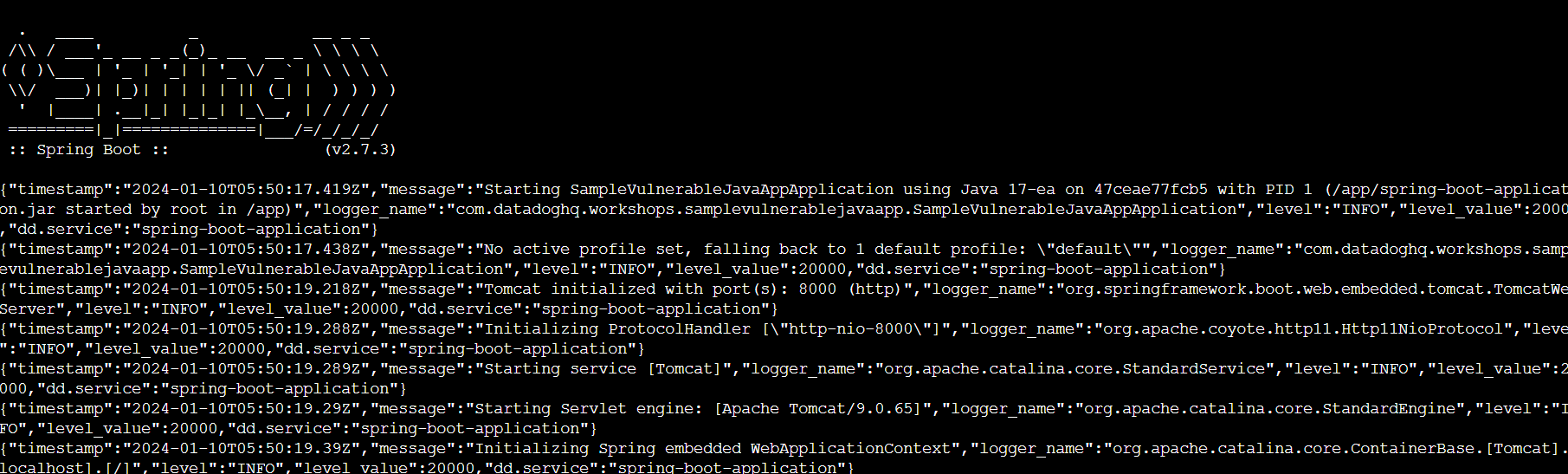
**Methodology:**

**Step 1:** create an Ec2 instance, install docker to that virtual machine. Make sure docker is running.

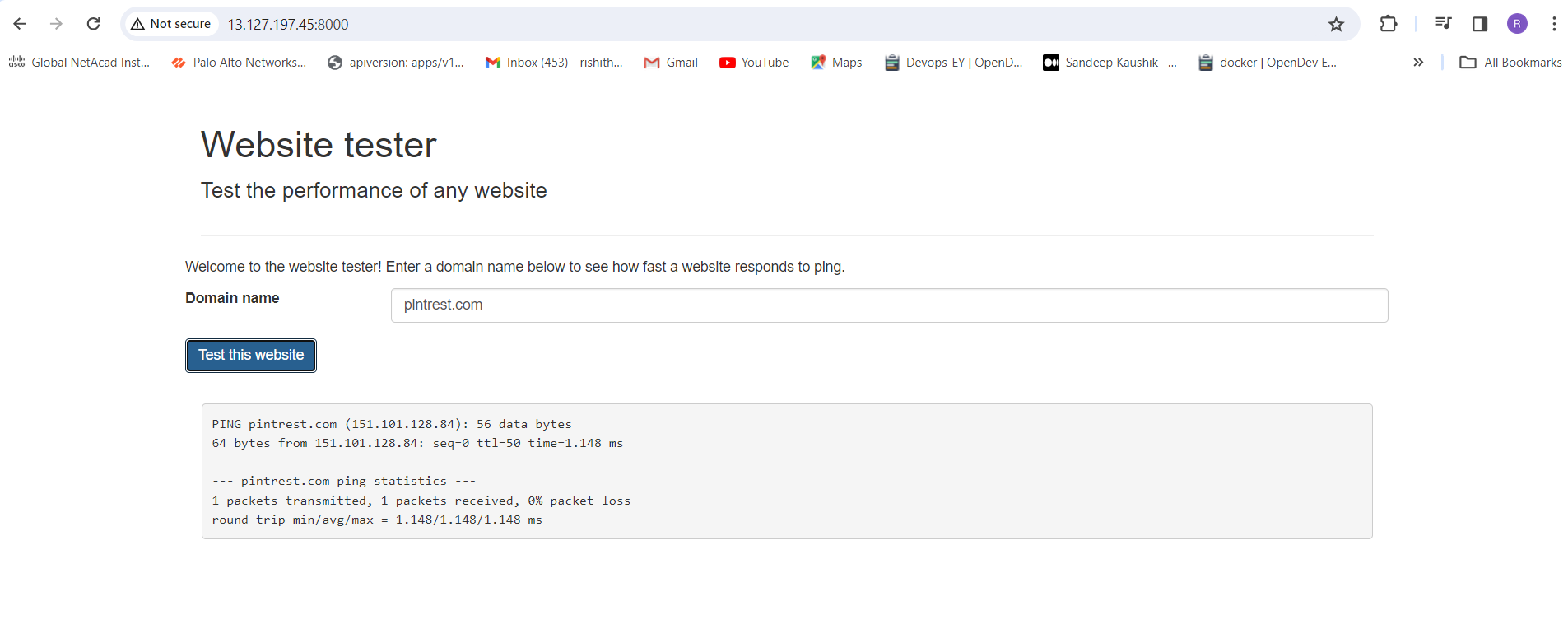


**Step 2:** run the following command to live that application





**Step 3:** live the application using the defined port



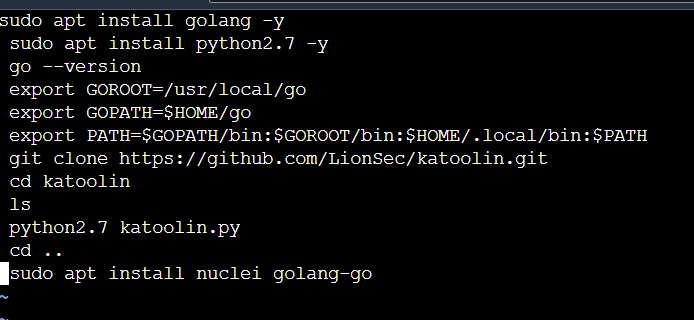
**Tools:**

1. Nuclei
2. Horusec
3. Synk

**Observation:**

**Analysis 1:** using Nuclei

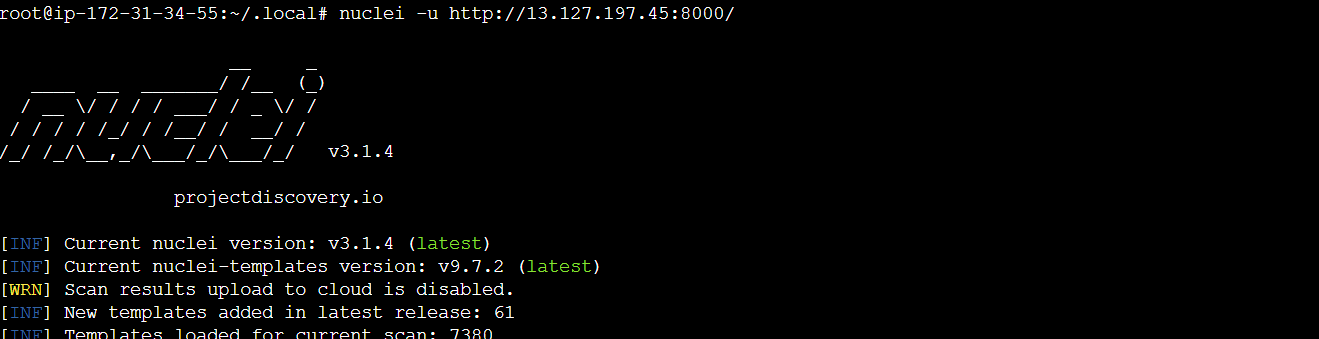
Steup the nuclei in your virtual machine using the following commands

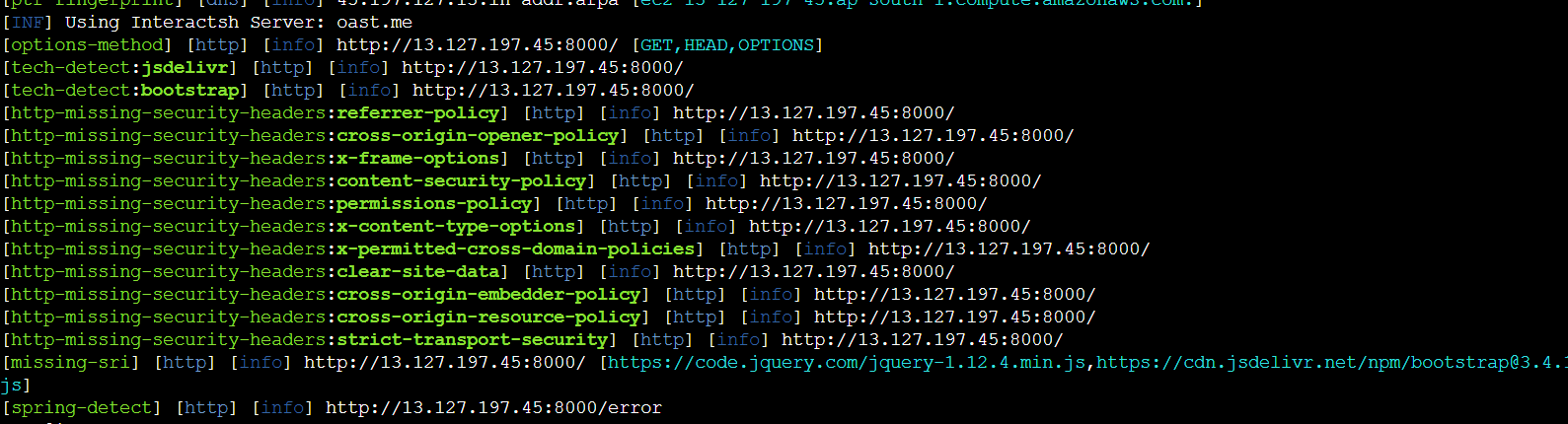


After setting up check nuclei is installed or not



Using nuclei command perform the test against the live application





The following are the some high vulnerabilities founded using nuclei

**Strict-Transport-Security:** The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) informs browsers that the site should only be accessed using HTTPS, and that any future attempts to access it using HTTP should automatically be converted to HTTPS.

**solution**: The HTTP Strict Transport Security header informs the browser that it should never load a site using HTTP and should automatically convert all attempts to access the site using HTTP to HTTPS requests instead.

**Cross-Origin Resource Policy (CORP):** Cross-Origin Resource Policy is a policy set by the Cross-Origin-Resource-Policy HTTP header that lets websites and applications opt in to protection against certain requests from other origins (such as those issued with elements like <script> and <img>), to mitigate speculative side-channel attacks, like Specter, as well as Cross-Site Script Inclusion attacks.

**solution:** The Cross-Origin-Embedder-Policy HTTP response header, when used upon a document, can be used to require sub resources to either be same-origin with the document, or come with a Cross-Origin-Resource-Policy HTTP response header to indicate they are okay with being embedded. This is why the cross-origin value exists.

**Clear-Site-Data:** The Clear-Site-Data header clears browsing data (cookies, storage, cache) associated with the requesting website. It allows web developers to have more control over the data stored by a client browser for their origins.

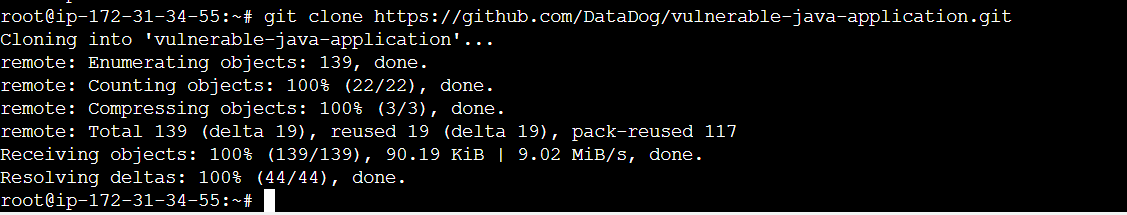
**Solution:** If a user signs out of your website or service, you might want to remove locally stored data. To do this, add the Clear-Site-Data header to the page that confirms the logging out from the site has been accomplished successfully.

**X-Content-Type-Options:** The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should be followed and not be changed. The header allows you to avoid MIME type sniffing by saying that the MIME types are deliberately configured.

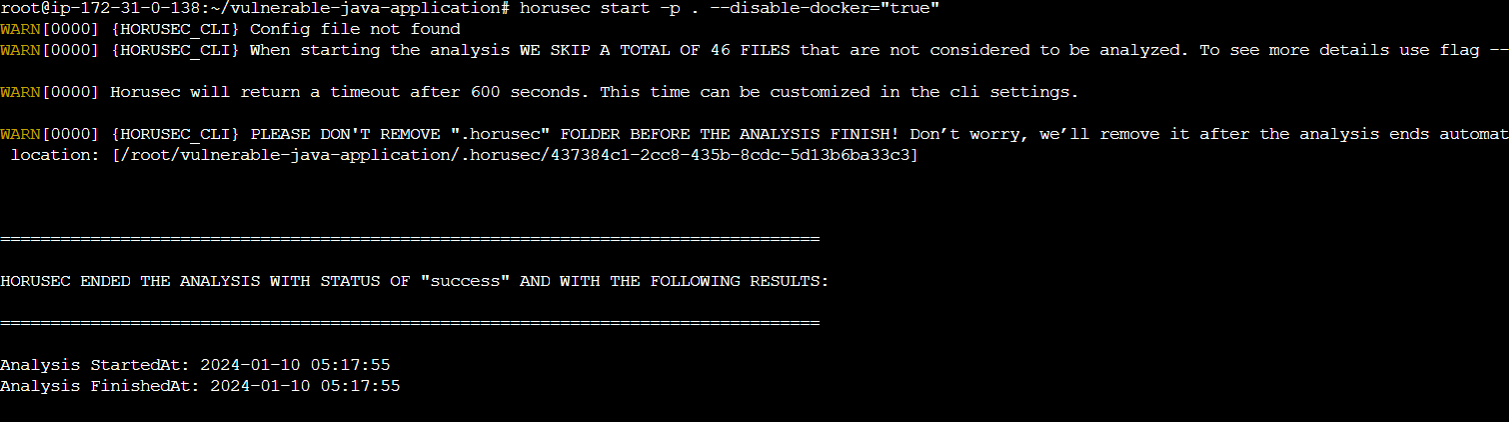
**Solution:** Blocks a request if the request destination is of type style and the MIME type is not text/css, or of type script and the MIME type is not a JavaScript MIME type.

**Analysis 2:** Using Horusec

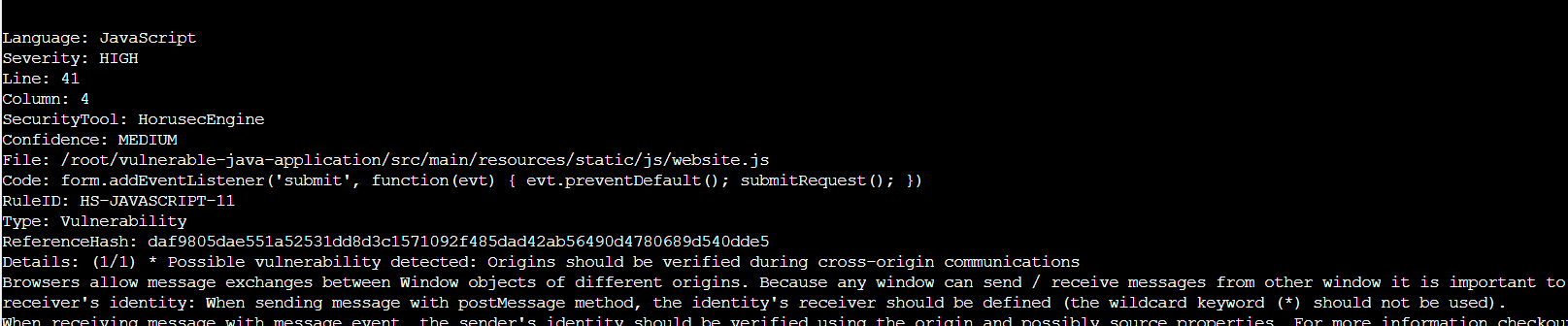
Clone the application in your local machine

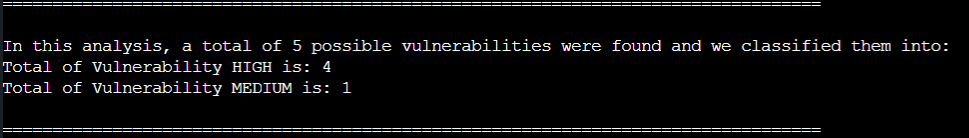


Then go into that directory perform the horusec scanning.



This command is used to scan vulnerabilities using horusec





In this git repo it find total of 5 vulnerabilities

High: 4

Medium: 1

The following are the details about the vulnerability and their mitigation process

**Possible vulnerability detected:** Origins should be verified during cross-origin communications

Browsers allow message exchanges between Window objects of different origins. Because any window can send / receive messages from other window it is important to verify the sender's / receiver's identity: When sending message with post Message method, the identity's receiver should be defined (the wildcard keyword (\*) should not be used).When receiving message with message event, the sender's identity should be verified using the origin and possibly source properties.

**Possible vulnerability detected:** Alert statements should not be used

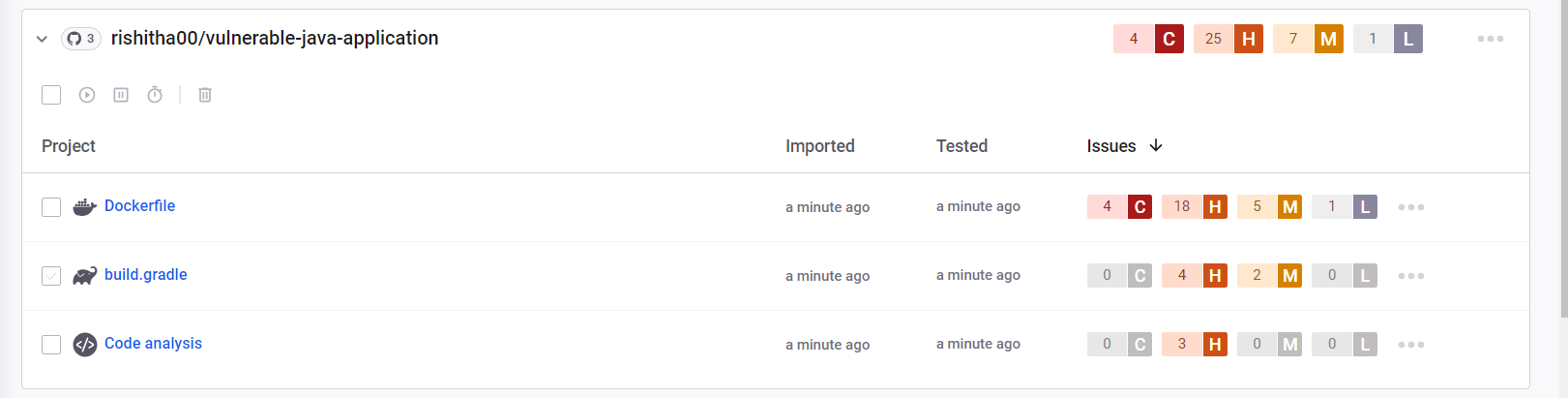
alert(...) as well as confirm(...) and prompt(...) can be useful for debugging during development, but in production mode this kind of pop-up could expose sensitive information to attackers, and should never be displayed.

**Possible vulnerability detected:** Origins should be verified during cross-origin communications

Browsers allow message exchanges between Window objects of different origins. Because any window can send / receive messages from other window it is important to verify the sender's / receiver's identity: When sending message with post Message method, the identity's receiver should be defined (the wildcard keyword (\*) should not be used).

**Analysis 3:** Using Snyk

Fork that data-dog-vulnerable-java-application in your repo , then perform the snyk



Critical: 4

High: 25

Medium: 7

Low :1

Found vulnerabilities in docker file , build.gradle and code analysis

The following are critical vulnerabilities with solution

**Classic Buffer Overflow:** The product copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.

**solution:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.For example, many languages that perform their own memory management, such as Java and Perl, are not subject to buffer overflows. Other languages, such as Ada and C#, typically provide overflow protection, but the protection can be disabled by the programmer.Be wary that a language's interface to native code may still be subject to overflows, even if the language itself is theoretically safe.

**Out-of-bounds Write:** The product writes data past the end, or before the beginning, of the intended buffer.

**solution:** Consider adhering to the following rules when allocating and managing an application's memory.Double check that the buffer is as large as specified.When using functions that accept a number of bytes to copy, such as strncpy(), be aware that if the destination buffer size is equal to the source buffer size, it may not NULL-terminate the string.Check buffer boundaries if accessing the buffer in a loop and make sure there is no danger of writing past the allocated space.If necessary, truncate all input strings to a reasonable length before passing them to the copy and concatenation functions.

**Off-by-one Error:** A product calculates or uses an incorrect maximum or minimum value that is 1 more, or 1 less, than the correct value.

**solution:** When copying character arrays or using character manipulation methods, the correct size parameter must be used to account for the null terminator that needs to be added at the end of the array. Some examples of functions susceptible to this weakness in C include strcpy(), strncpy(), strcat(), strncat(), printf(), sprintf(), scanf() and sscanf().

**Loop with Unreachable Exit Condition:** The product contains an iteration or loop with an exit condition that cannot be reached, i.e., an infinite loop.

**solution:** method processMessagesFromServer attempts to establish a connection to a server and read and process messages from the server. The method uses a do/while loop to continue trying to establish the connection to the server when an attempt fails.