Table of Contents

[1. Background on Semaphore API 2](#_Toc163663237)

[2. Test Approach 2](#_Toc163663238)

[3. Test Case Design 2](#_Toc163663239)

[4. Test Case Naming Convention. 3](#_Toc163663240)

[5. Test Environment: 5](#_Toc163663241)

[6. Test Execution: 13](#_Toc163663242)

[7. Test Monitoring and Reporting 16](#_Toc163663243)

[8. Performance Test Tool As JMeter. 18](#_Toc163663244)

**This Document provides the details about how to Test Semaphore API Automation Using JMeter.**

API testing is Performed based on set of documents starting from Test Cases, Test plan/Test strategy, Traceability matrix,

# Background on Semaphore API

The Semaphore API is locked to apply on the process running in Front End Turbine, there will be concurrent process running in Front End Turbine, to set a lock to the concurrent process in turbine front end using API Calls so Semaphore API is used, this API operation allows users to explicitly lock a resource to prevent concurrent processing. There is API Methods which allow for acquiring and releasing a semaphore.

# Test Approach

1. Test cases will be created and refreshed whenever there are changes related to semaphore API.
2. Tests will be performed whenever there are changes that relate to semaphore components, this includes changes in semaphore API and semaphore database.
3. Tests will be performed as part of regression test to ensure semaphore is functional.
4. The Semaphore API for Automated testing will be using JMeter Automation tool.

# Test Case Design

1. In this stage Test cases are Written in Azure test plans for Semaphore API Features.
2. All the test cases are Mapped to Work items in azure devops.

**All Test Scenarios identified as of now.**

1. GET-Semaphore Scenarios

Fetch the semaphores.

1. POST – Semaphore scenarios.
   1. acquire a lock if no blocking semaphores exist.
   2. reject when acquiring and another exclusive semaphore exist for different process key.
2. PUT-Release – Semaphore scenario s.

should release semaphore.

1. PUT-Renew

Should renew semaphore.

# Test Case Naming Convention.

Below are the naming conventions followed by the QA team in azure test plan.

TCXXX\_Product\_Feature\_Functionality\_TestcaseName

Eg : TC001\_Metadata\_SemaphoreAPI\_AquireLock\_to Validate that lock is acquired when Semaphore is not Blocked.

Testcase Naming convention should follow as Testcase Id followed by Product Name, Feature Name, Functionality, Test Case Objective.

**Top 2 - Test Cases:**

**TC#1**

TC001\_Metadata\_SemaphoreAPI\_AquireLock\_to Validate that lock is acquired when Semaphore is not Blocked.

**Objective:** to Validate that lock is acquired when Semaphore is not locked.

**Precondition:**

1- All access, Bearer token and authentication should be available   
2- there should not be any exiting lock on semaphores in the system for the process to acquire lock.

**Steps:**

1.With API Post " POST URL" request the lock

2.Validate the DB table Expected.

**expected Result:**

1.the lock should be acquired successfully.

2.the system should indicate that the lock has been acquired.

**Actual Result:**

**TC#2**

TC002\_Metadata\_SemaphoreAPI\_RejectLock\_to Validate that lock is Rejected when Semaphore is Blocked.

**Objective:** to Validate that lock is rejected when Semaphore is locked

**Precondition:**

1- All access, Bearer token and authentication should be available   
2- there should not be exiting lock on semaphores in the system for the process to reject lock.

**Steps:**

1.With API Post " POST URL" request the access to acquire lock

2.Validate the DB table Expected.

**expected result:**

1.the lock should be rejected.

2.the system should indicate that the lock has been rejected.

**Actual Result:**

A diagram of a computer

Description automatically generated

The above screen shows the test plan structure.

The same structure is maintained in azure test plan.

A screen shot of a computer

Description automatically generated

The above Screenshot shows the Test plan structure maintained in Azure Test Plan.

A screenshot of a computer

Description automatically generated

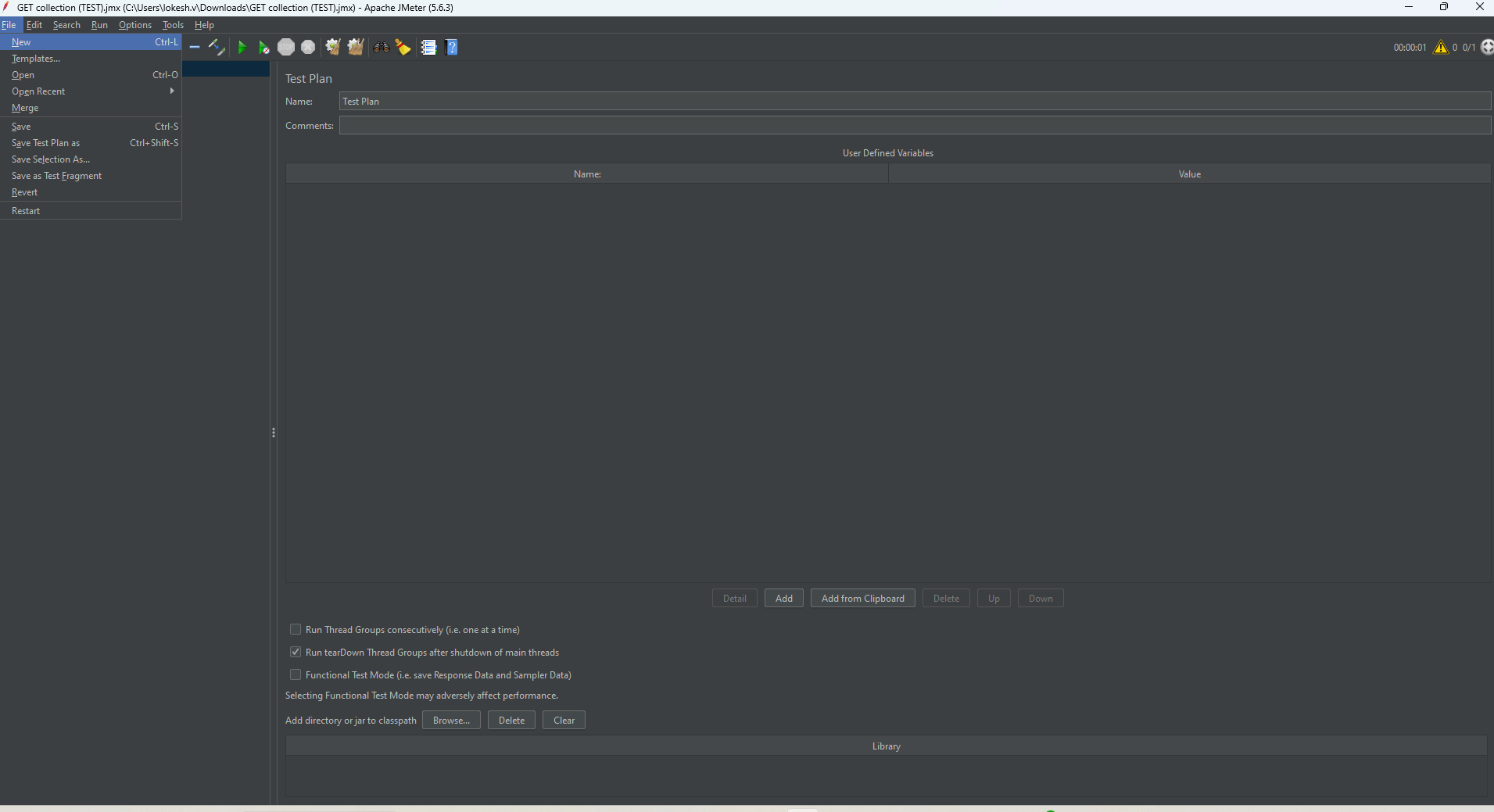
The above screenshot shows the linking of the test plan to the PBI or work item and ensures traceability is maintained.

# Test Environment:

In this stage we will prepare the test environment using the JMeter tool

Below are the screen shots step by step to set up the JMeter tool for Test case execution.

Belo snapshot shows how to add the Test plan, click on File 🡪 Select New



Right click on Test plan and select Add 🡪 Threads (Users)🡪 Thread Group

A screenshot of a computer

Description automatically generated

Below snapshot shows how to Rename as Semaphore API

A screenshot of a computer

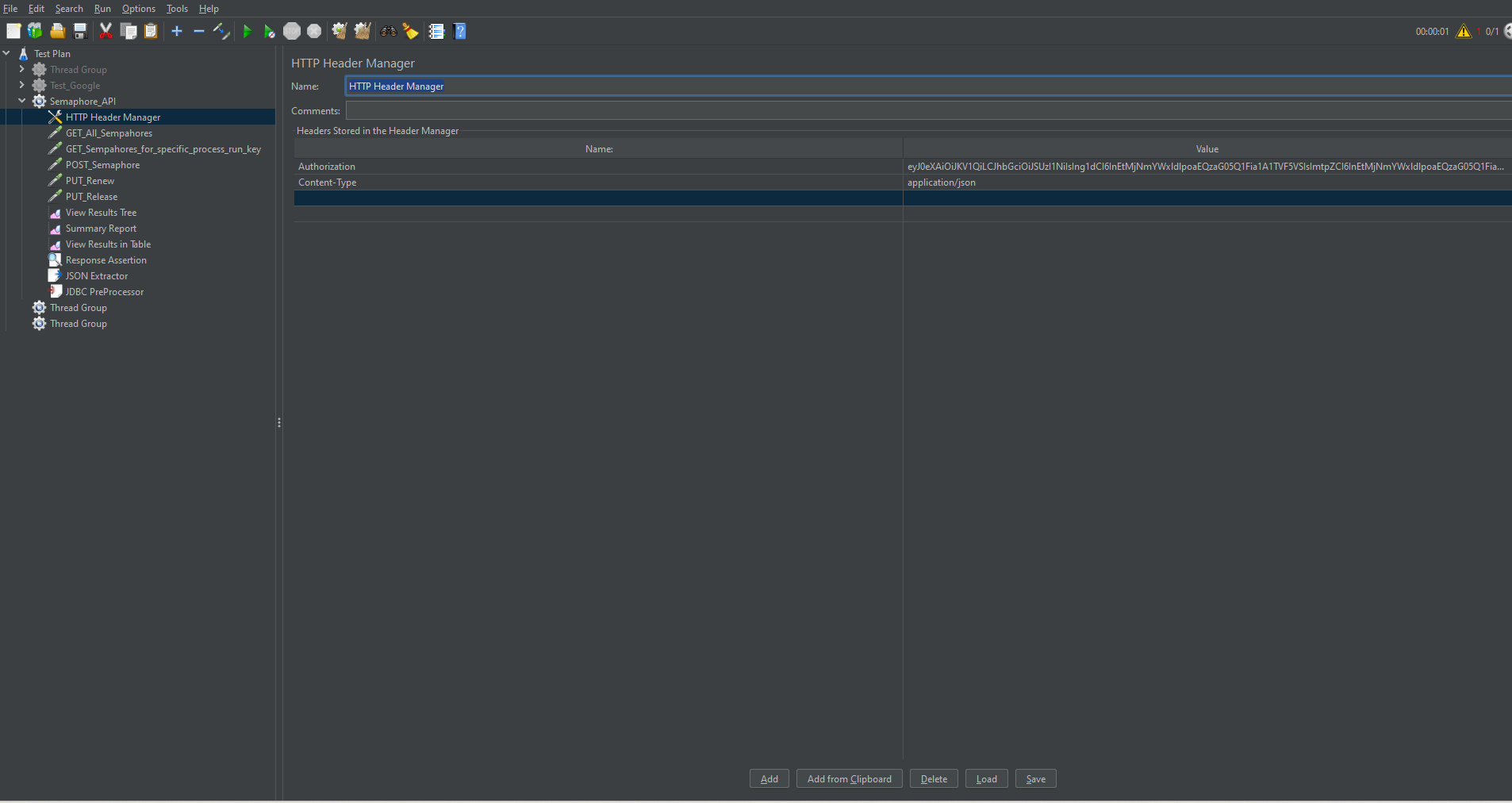
Description automatically generated

Right Click on Semaphore API and select Add 🡪 Sample🡪 HTTP Request

A screenshot of a computer

Description automatically generated

Rename the HTTP Request as HTTP Header Manager



Add the Authorization and content type as Parameters.

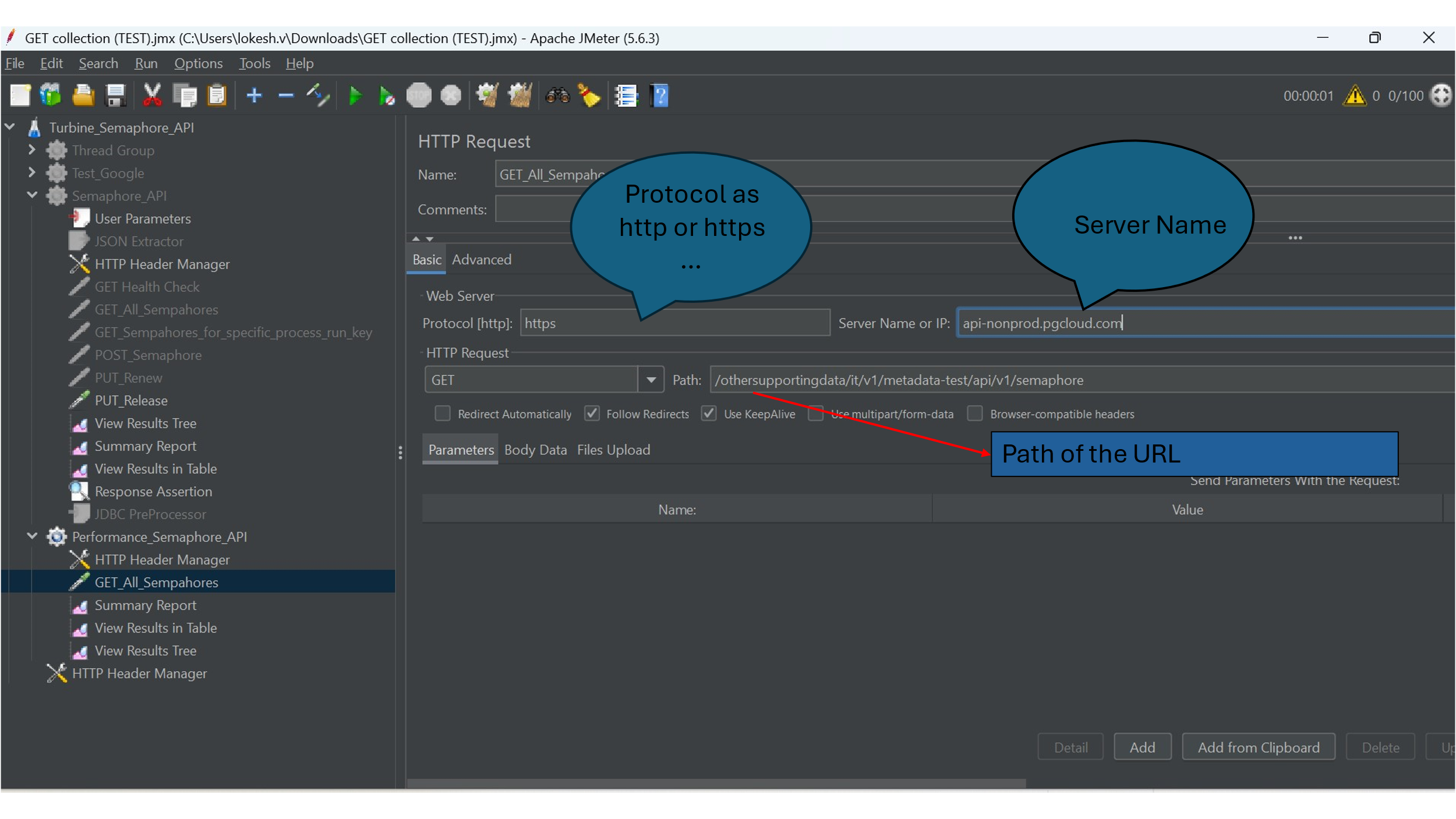
A screenshot of a computer

Description automatically generated

Right Click on Semaphore API and select Add 🡪 Sample🡪 HTTP Request and Rename as GET\_ALL\_Semaphores.

A screenshot of a computer

Description automatically generated

Right Click on Semaphore API and select Add 🡪 Sample🡪 HTTP Request and Rename as GET\_Semapshore\_for\_specific\_process\_run\_key and provide the parameters as process\_run\_key as mandatory field and optional as resourcePath. 

# Test Execution:

In This Stage We run the test case and validate the API test results for expected against the Actual results.

Here are a few examples of semaphore API methods used where in client send the request and server respond.

1. GET Request for all semaphores for 20 users(threads), below is the snapshot for how to configure for 20 users.
2. Click Save
3. Clear all the results and click Run!

# Test Monitoring and Reporting

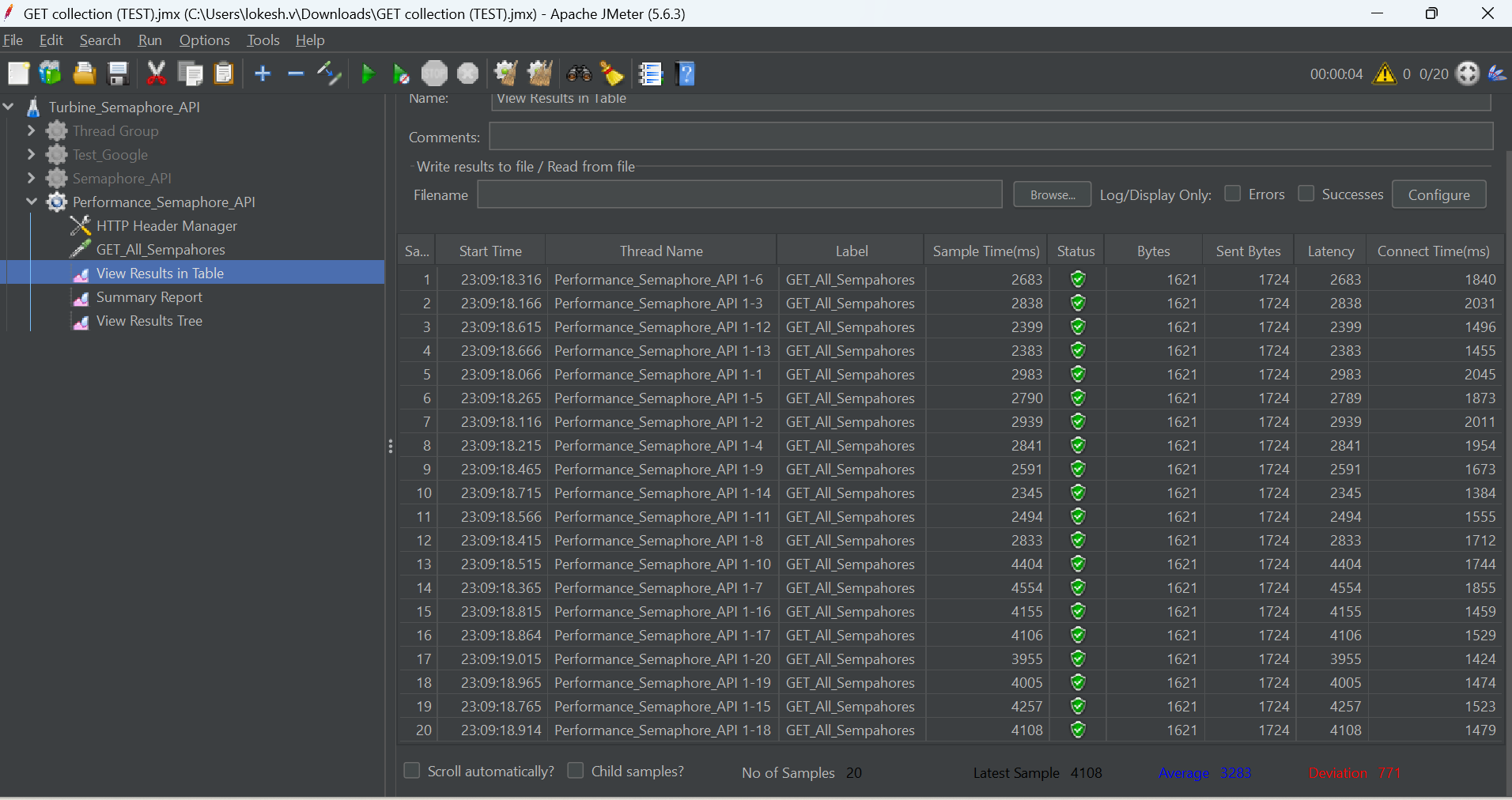
This is the final stage where we monitor the results and report the test results to the stakeholders through email. We generate the detailed reports from the collections in the post man including any issues/findings.

Below snapshot is the setup for 20 users, to test the performance.

A screenshot of a computer

Description automatically generated

Below is the snapshot result for 20 users in View Result in Table.



Below showing the snapshot for Summary Report 20 users.

A screenshot of a computer

Description automatically generated

Below snapshot shows the sample result for each individual request.

A screenshot of a computer

Description automatically generated

Below is the snapshot for 20 users results in View Results in Tree format of Request Body.

A screenshot of a computer

Description automatically generated

Below is a snapshot of the Response body for 20 users in View Result Table.

A screenshot of a computer

Description automatically generated

Below snapshot is the setup for 40 users, to test the performance.

A screenshot of a computer

Description automatically generated

Below is the snapshot of Response body for 40 users in View Result Table.

A screenshot of a computer

Description automatically generated

Below showing the snapshot for Summary Report 40 users.

A screenshot of a computer

Description automatically generated

Below snapshot shows the sample result for each individual request.

A screenshot of a computer

Description automatically generated

Below snapshot shows the sample request body for each individual request.

A screenshot of a computer

Description automatically generated

Below snapshot shows the sample response data for each individual request.

A screenshot of a computer

Description automatically generated

Below snapshot is the setup for 60 users, to test the performance.

A screenshot of a computer

Description automatically generated

Below is the snapshot of Response body for 60 users in View Result Table.

A screenshot of a computer

Description automatically generated

Below shows the snapshot for Summary Report 60 users.

A screenshot of a computer

Description automatically generated

Below is the snapshot of Response body for 80 users in View Result Table

A screenshot of a computer

Description automatically generated

Below shows the snapshot for Summary Report for 80 users.

A screenshot of a computer

Description automatically generated

Below is a snapshot of Response body for 100 users in View Result Table.

A screenshot of a computer

Description automatically generated

Below snapshot is the setup for 100 users, to test the performance.

A screenshot of a computer

Description automatically generated

Below is the snapshot for 100 users, the snapshot shows the view result in table format.

A screenshot of a computer

Description automatically generated

Below shows the snapshot for Summary Report for 100 users.

A screenshot of a computer

Description automatically generated

# Performance Tool Integration with CI/CD

JMeter has documented the integration with Jenkins here Jenkins is used for example purposes only, we can integrate with any CI/CD tool. We just need to enable the plugin and follow the steps to achieve the integration. [Using JMeter with Jenkins](https://www.jenkins.io/doc/book/using/using-jmeter-with-jenkins/)



# Performance Test Tool as JMeter.

How to use the tool to support testing please follow the below link

<https://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html>