Getting Started with Taurus

Lab Guide

BlazeMeter

- PROPRIETARY AND CONFIDENTIAL INFORMATION -

Table of Contents

Lab – Downloading and Installing Taurus

Goals	Download and install Taurus, a test automation tool.
Scenario	Taurus is a free and open source automation framework, which is basically an abstraction layer over JMeter, Grinder, Gatling, and Selenium. Taurus provides an easily-readable, version control friendly and unified DSL (Domain-specific language) mechanism to define load test scenarios. There are different ways to install Taurus on your windows system. In this demonstration, we will use the prebuilt installer that installs Python and the latest Taurus version on your local system. In case, you already have Python installed on your system; you can manually install the Taurus using PIP (Python Package Manager). In this demonstration, you will: ▪ Download the installer ▪ Validate that Taurus has been successfully installed
Time	10 minutes

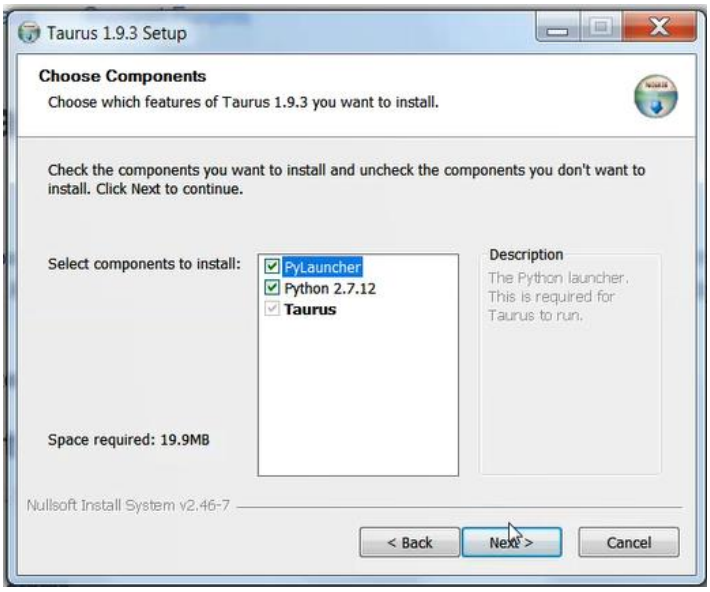
Instructions:

1. Enter gettaurus.org in your browser.
2. In the web page, click the **Installation** tab. You will be redirected to the **Installing and Upgrading** page that provides you the information for installing Taurus.
3. Since most users have the Windows operating system, let's download the windows installer. Download the latest Windows installer from the website.
4. Run the installer and wait until the Installation wizard opens.

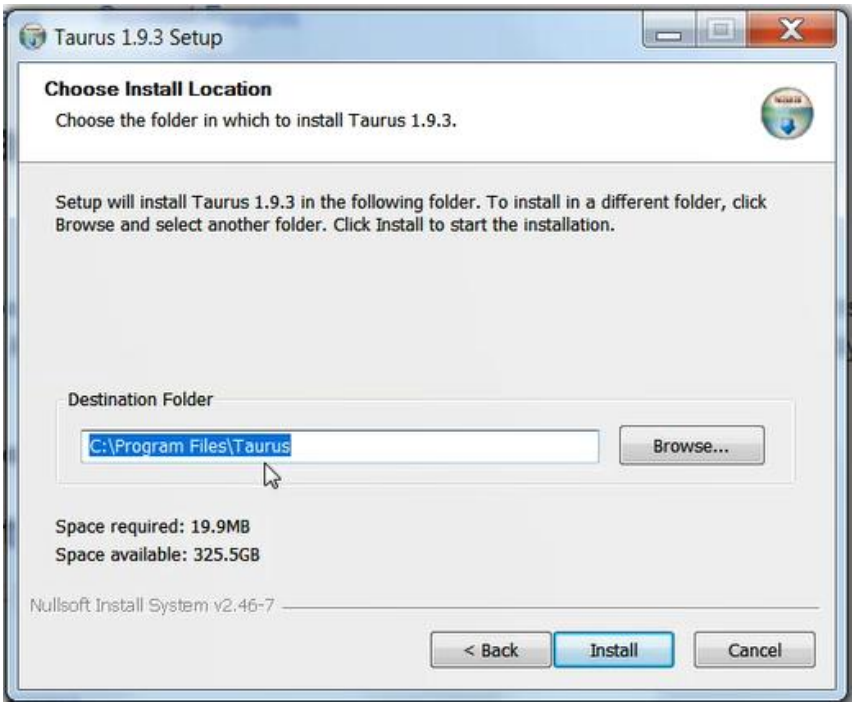
5. Click **Next** to continue the installation process.

6. Select the components that you want to install and click **Next**. It will install the following components:

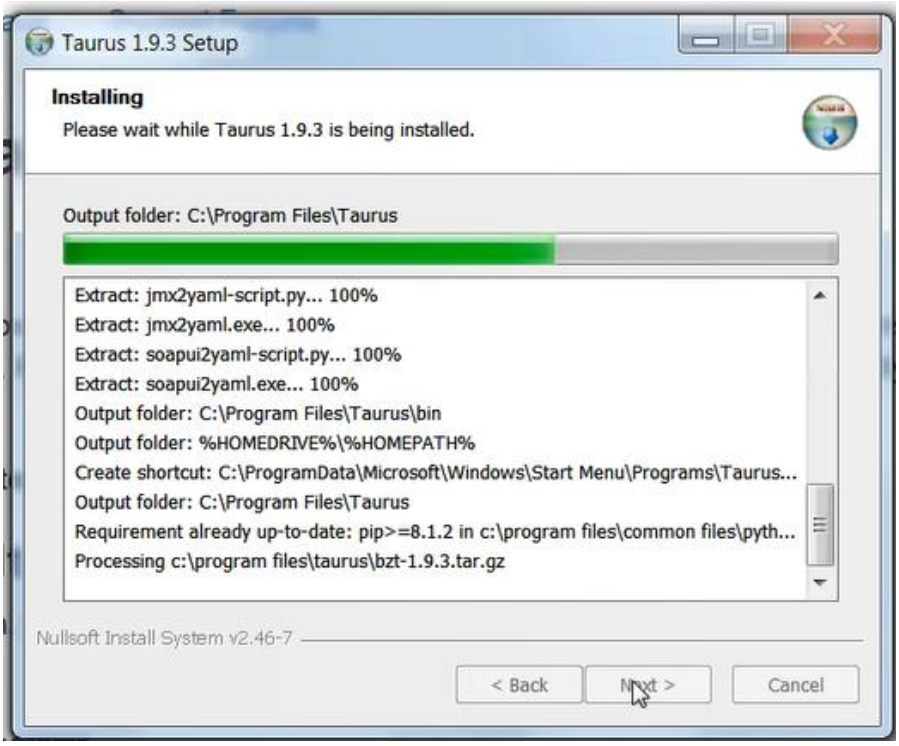
    i. Python 2.7
    ii. PyLauncher needed to launch Python programs
    iii. Taurus



7. You can view the default location where Taurus will be installed. To change the default location, click **Browse** and navigate to the target destination folder. Click **Install** to start the installation.

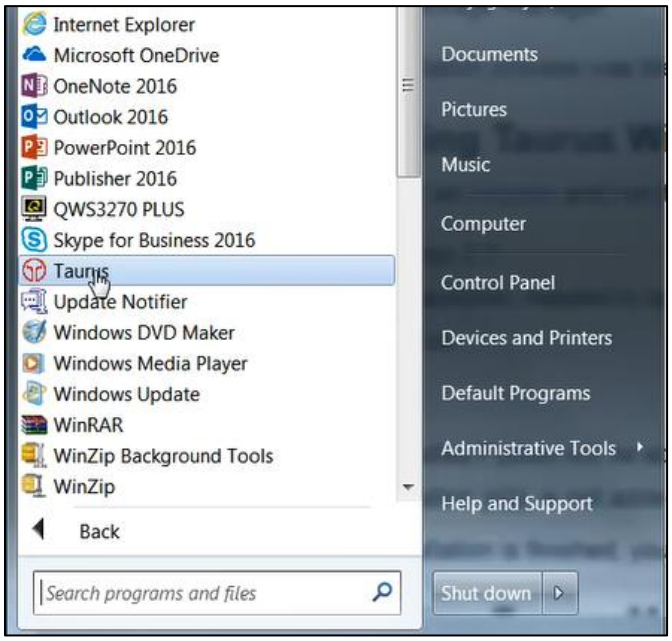


8. The relevant files are extracted, and the Taurus is installed on your system.



9. Taurus is successfully installed. Click **Finish** to close the installation wizard.

10. Click **Start** button from the taskbar and check for Taurus in **All Programs** pop-up.



11. Follow the instructions from http://gettaurus.org/install/Installation/ to install Taurus on other operating systems such as Mac OS and Linux (Ubuntu and CentOS). This completes the Taurus download and installation on your local system.

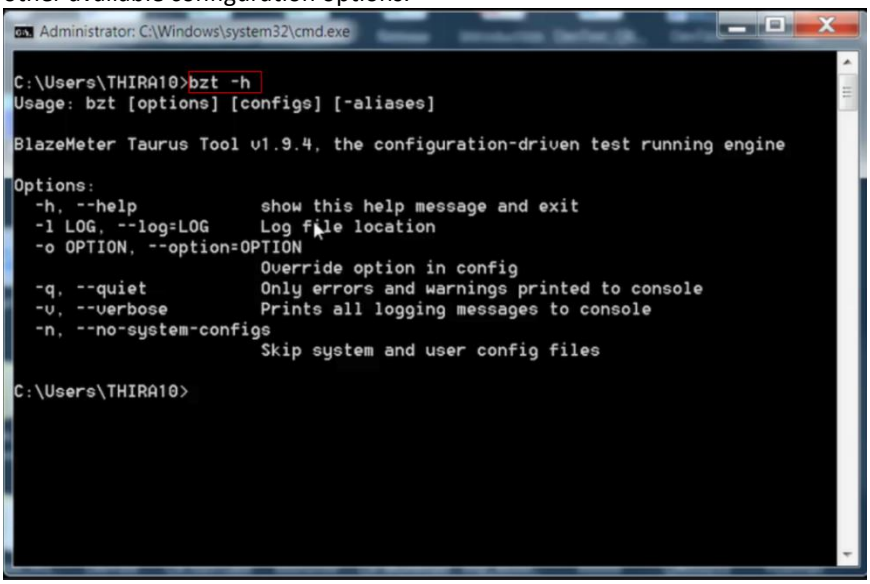
Lab – Executing Basic Taurus Commands

Goals	Running Taurus and exploring the basic Taurus command-line options.
Scenario	In this lab, you will learn the basic command-line options available for Taurus. You will run the bzt command to view the current Taurus version installed on your local system. Further, you can navigate to the artifacts directory to view the log file and configuration files that are created by default. In this demonstration, you will: ▪ Explore the Taurus command line options ▪ Navigate to Artifacts directory ▪ Run the bzt command ▪ Review the configuration and log files
Time	10 minutes

Instructions:

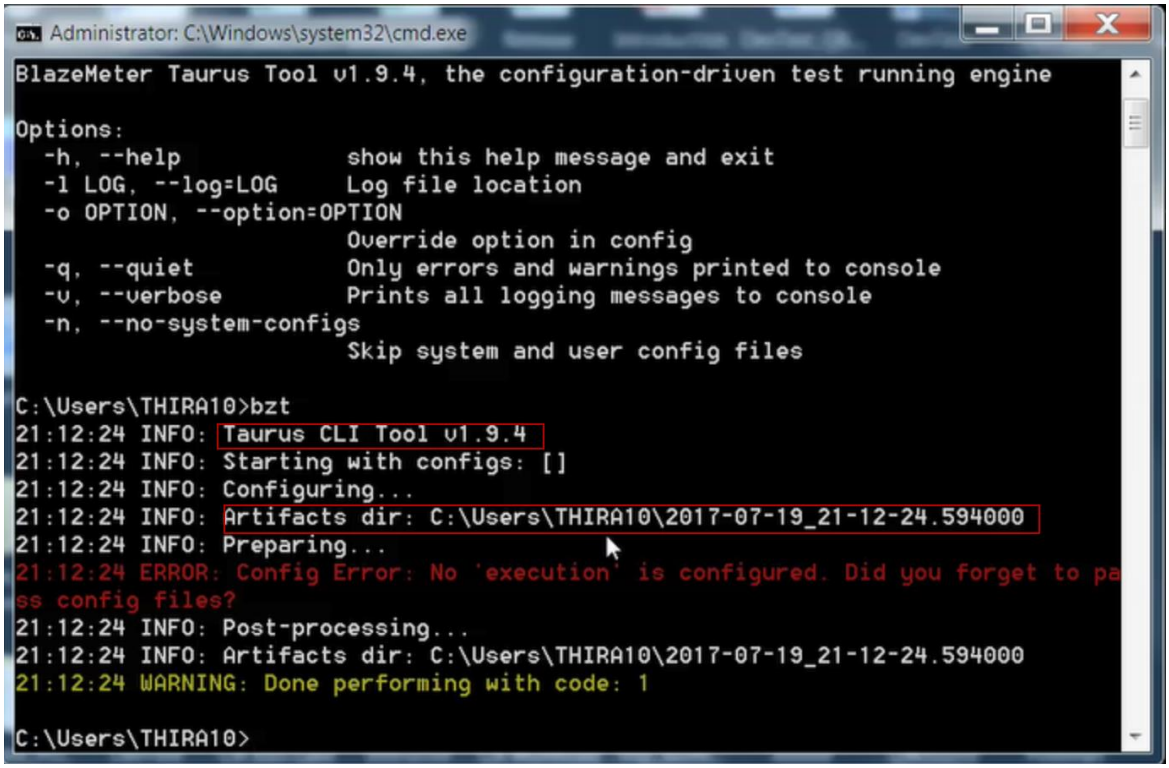
1. Taurus is a command-line tool. It is executed by using the bzt command. Note that bzt is an acronym for BlazeMeter Taurus.

2. To view the Taurus version installed on your local system, enter **bzt -h** in the command-line. Press Enter to view other available configuration options.



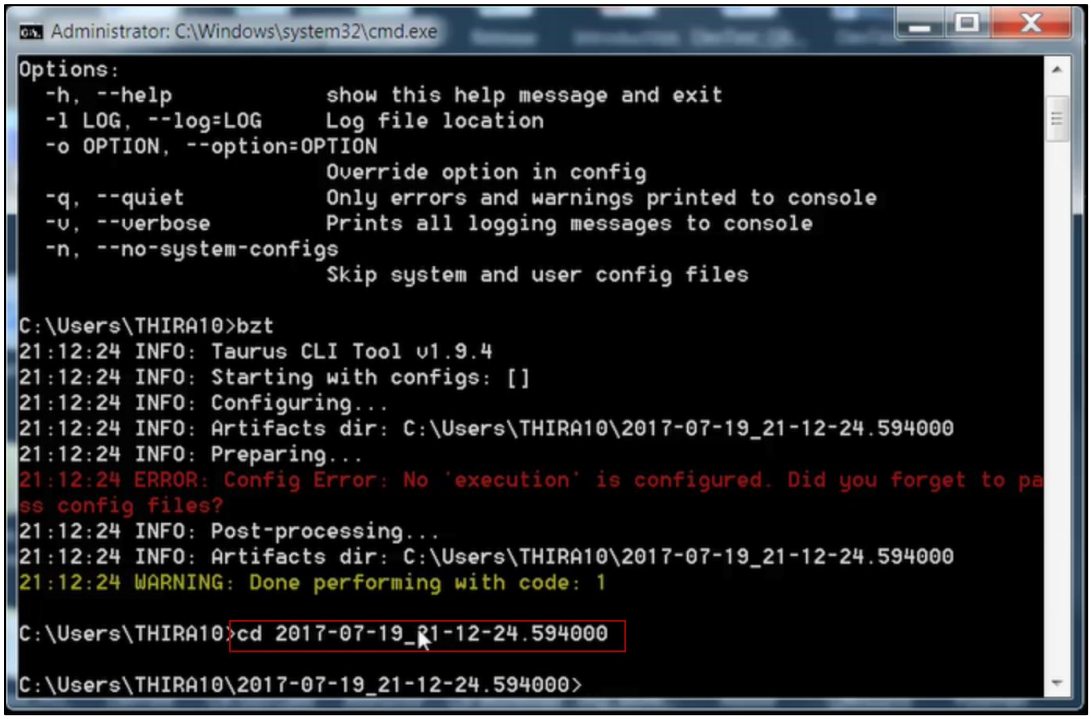
Options	Description
-h, --help	This option shows help message and exit.
-q, --quiet	This option shows only errors and warnings printed to console.
-n, --no-system-configs -	This option skips the system and user configuration files (it means that no system configuration files will be used during the execution).
-v, --verbose -	This option allows you to print all logging messages to console.
-l LOG, --log=LOG –	This option allows you to change log file location. By default, **bzt.log in** is the current log directory.
-o OPTION, --option=OPTION	This option allows you to override some of the default options in your configuration file. It can be used multiple times.

3. Enter **bzt** in the command-line and press **Enter**. You could observe the information related to Taurus command line, config files, and artifacts directory. The Artifacts directory primarily collects all files that are used with execution: configs (except personal), logs, and generated scripts. The artifacts directory is mapped to the folder where you have installed Taurus, and it is represented with the current date and time stamp.

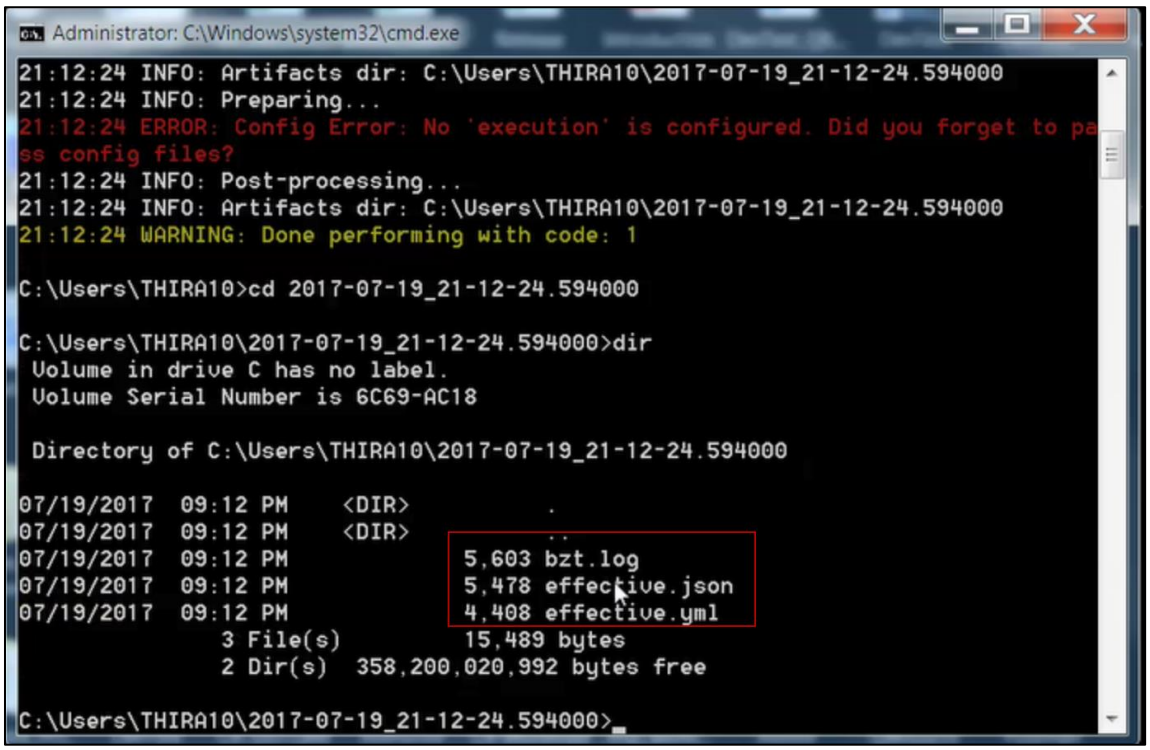


4. Browse to the directory where the Taurus file is installed. To change the directory, type **cd 2017-07-19_21-12-24.59400** and press **Enter**.

Note: The aritifacts directory created for you will have the associated time stamp settings as referenced in the screenshot. Please use that value to navigate to the artifacts directory.

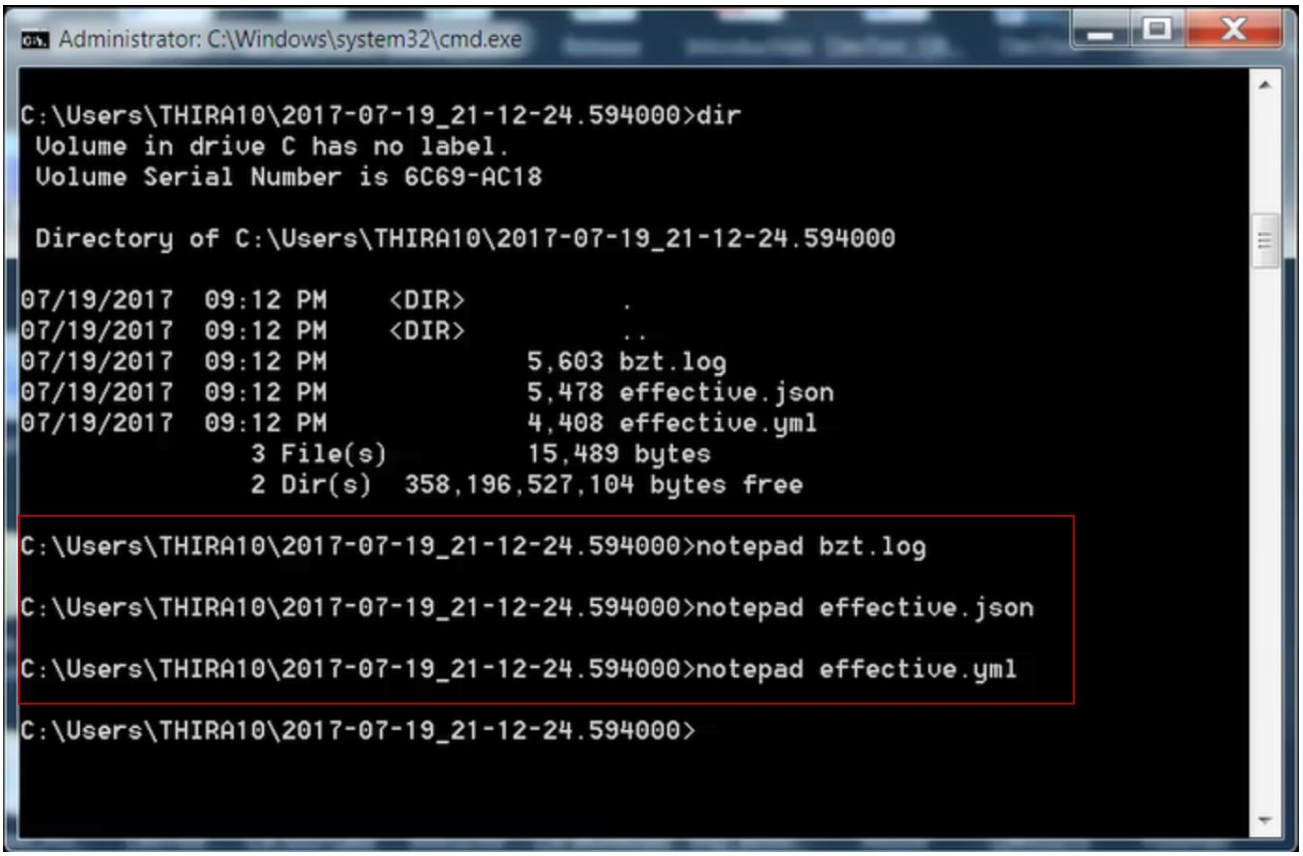


5. Enter **dir** command to view the files available in the directory. You could observe the three files **bzt.log**, **effective.json,** and **effective.yml**.



File name	Description
Bzt.log	Taurus log serves as a great source for troubleshooting the tool by finding the errors.
Effective.json and effective.yml	Configuration files after you apply defaults, shorthand rules, and any other modifications during execution are always saved in two formats.

6. To open the respective files, type the commands **notepad bzt.log**, **notepad effective.json** and **notepad effective.yml**. Review the files and close them.



7. This concludes this demonstration.

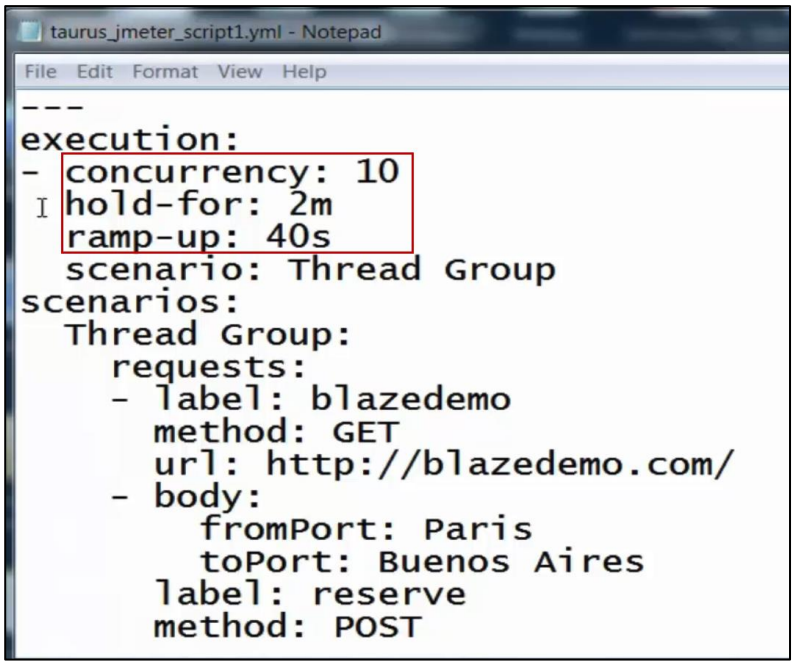
Lab – Execute a Taurus YAML Script

Goals	Execute a Taurus YAML Script.
Scenario	When it comes to performance testing, JMeter is an excellent tool, but it's challenging when you try to automate and integrate with other systems such as Jenkins, Selenium, Dynatrace APM, and other performance testing tools. JMeter's limited reporting capabilities, complex XML code, and its indefinite approach to set the test configuration details such as ramp-up, ramp-down, and concurrency makes the load testing process more complicated. To overcome these challenges, BlazeMeter introduced Taurus (Test Automation Running Smoothly), an open source test automation tool that abstracts and extends JMeter. Taurus script is written in YAML (a human-readable data serialization language), it makes the YAML configuration files that can be easily interpreted by someone who does not have much experience with any of the testing tools. In this lab, we will learn how to write a simple YAML script, execute it using the basic bzt command, and view the results in Taurus console for better analysis. You will accomplish this by completing the following tasks: ▪ Writing a simple YAML Script ▪ Modify the parameters ▪ Execute the script and view results in Taurus console
Time	10 minutes

Instructions:

Download the lab resources and save it on your local system. You can directly use these JMX, YAML and Python scripts to run the labs in future.

1. Open the **taurus_jmeter_script1.yml file** from your resource folder available on your local system. The YAML Script used here defines that there will be 10 virtual users and the execution will run for 2 minutes. The ramp-up defines that after 40 seconds all the virtual users will be available for execution.

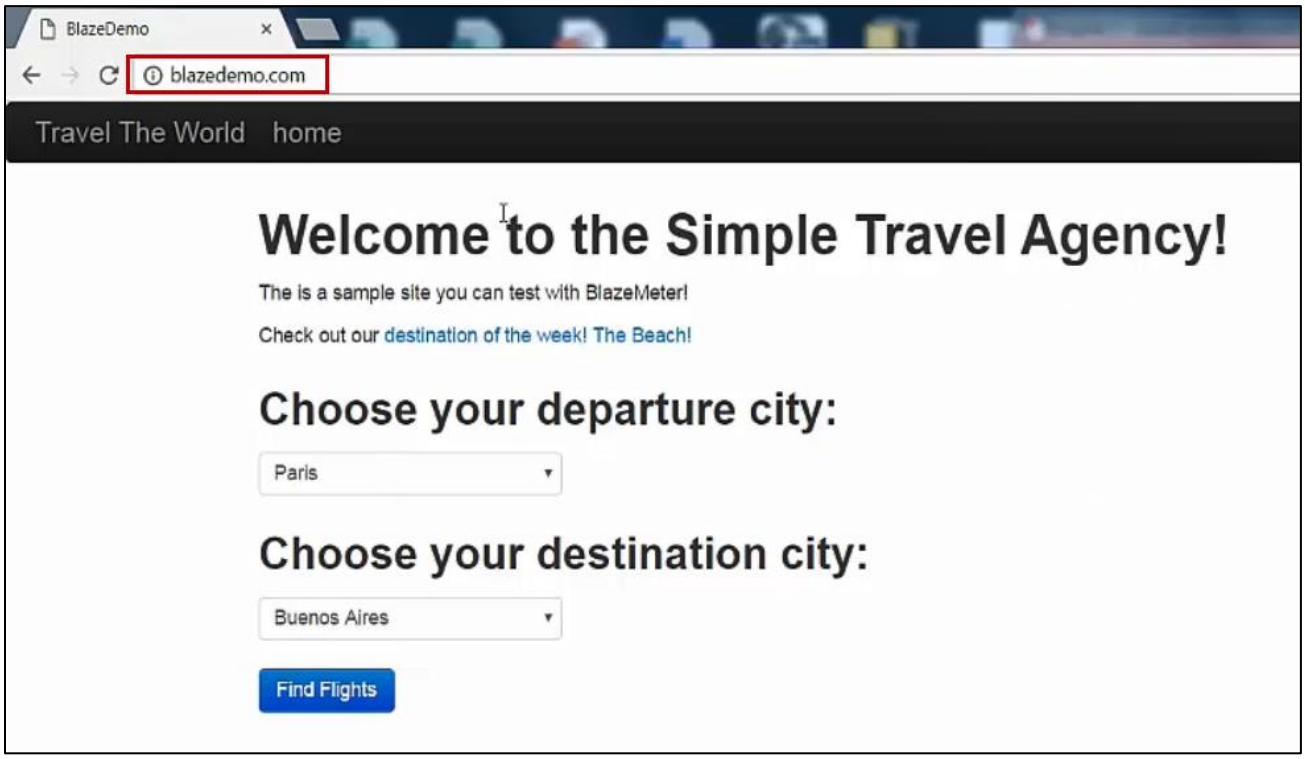


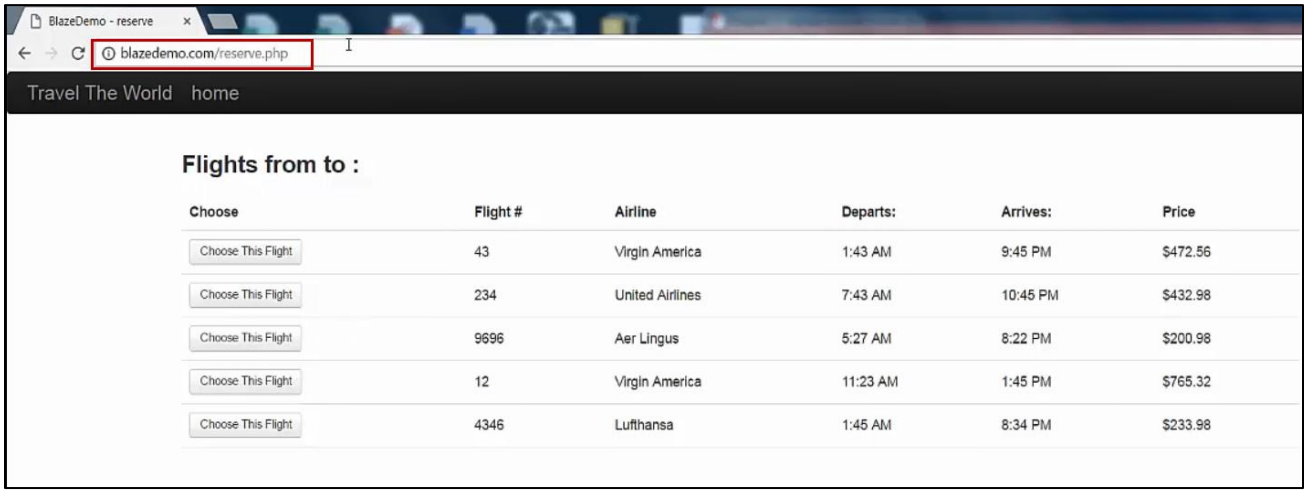
Field	Description
Concurrency	The number of target concurrent virtual users.
Hold-for	The time to reach target concurrency.
rampup	The time to hold target concurrency.

Note: As no executor is specified in the script, Taurus will use the default executor, JMeter.

2. The websites that you invoke by running the YAML script are:

   - http://blazedemo.com
   - http://blazedemo.com/reserve.php





3. Execute the Taurus script YAML from the command-line terminal using the command: **bzt taurus_jmeter_script1.yml**

4. Press Enter, the execution starts, and you can see the Taurus console.

5. From the console, you could observe the active users, latest interval stats, the time taken to connect to the request, latency in the network and, the number of bytes received. It even displays the response code of the website and the number of hits on each website that you have entered in the script along with errors per hits.

6. This concludes the lab exercise.

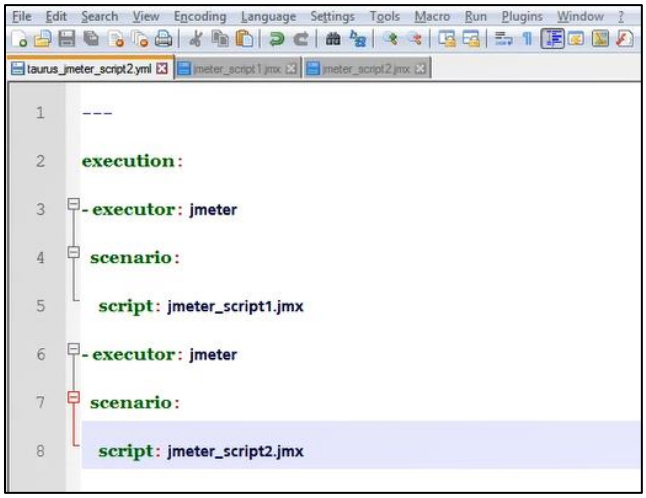
Lab – Execute a JMeter Script Using Taurus

Goals	Execute a JMeter Script using Taurus
Scenario	When it comes to performance testing, JMeter is fantastic, but not perfect. Automation and integration with other systems can be a pain, and the tool itself comes with a steep learning curve. Hence, BlazeMeter introduced Taurus, a free and open source automation framework, which is basically an abstraction layer over JMeter. Taurus provides a simple way to create, run and analyze performance tests. In this demonstration, we will parameterize multiple .jmx files by using a YAML script. You can also use YAML to modify JMX scripts. When running JMeter through Taurus, you can: • Run an existing JMeter script • Create a new JMeter script very easily using a YAML text file In this demonstration, you will: ▪ Write a simple YAML script ▪ Execute the YAML script by passing JMeter test scripts ▪ View results in Taurus console ▪ Run -gui option ▪ Run multiple .jmx files parallelly ▪ Generate a report using -report option for a simple jmx file
Time	10 minutes

Instructions:

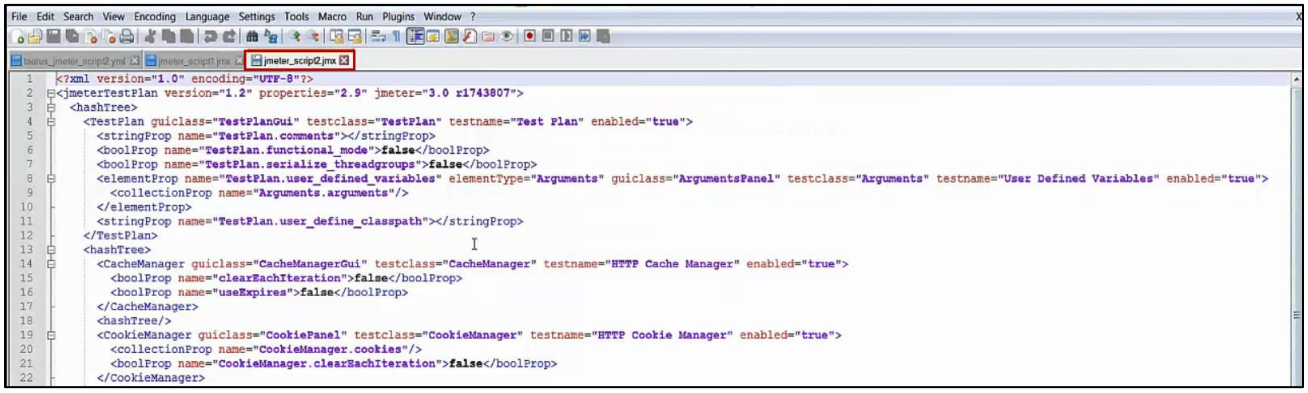
Download the lab resources and save it on your local system. You can directly use these JMX, YAML and Python scripts to run the labs in future.

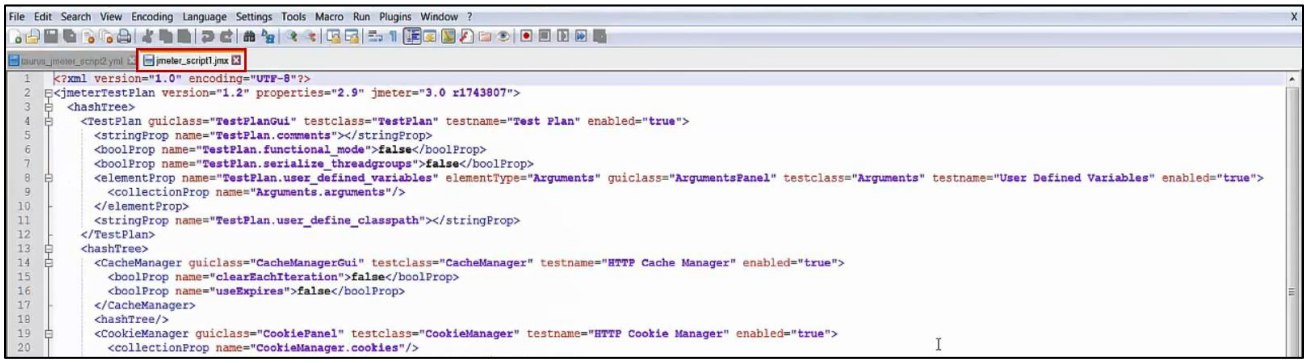
1. Open the **taurus_jmeter_script2.yml** file from your lab resources folder available on your local system.



2. Using Notepad++, you can open the JMeter scripts. Enter the following commands in command-prompt line to view the respective JMeter scripts.

   - **Notepad++ jmeter_script1.jmx**
   - **Notepad++ jmeter_script2.jmx**

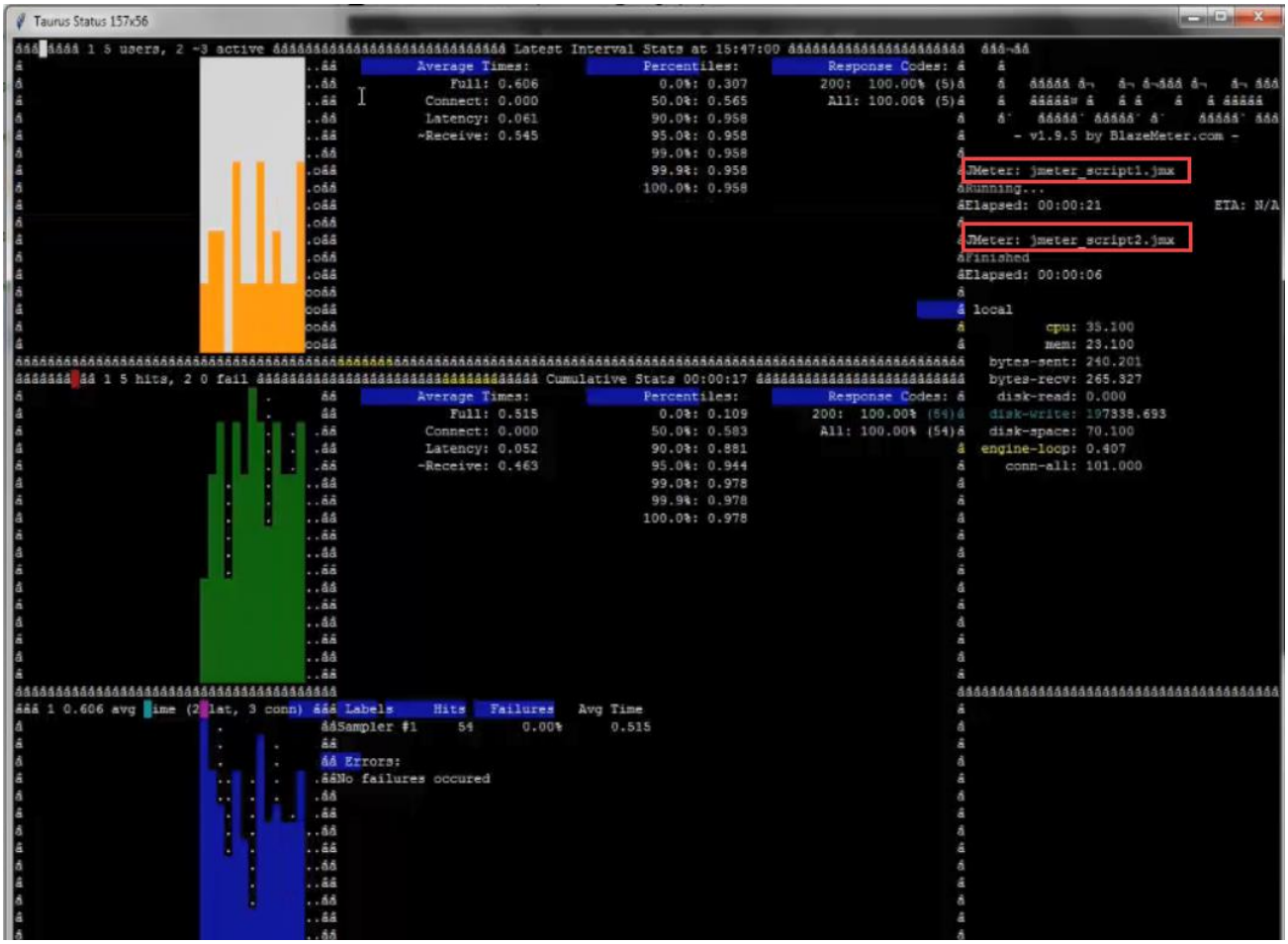




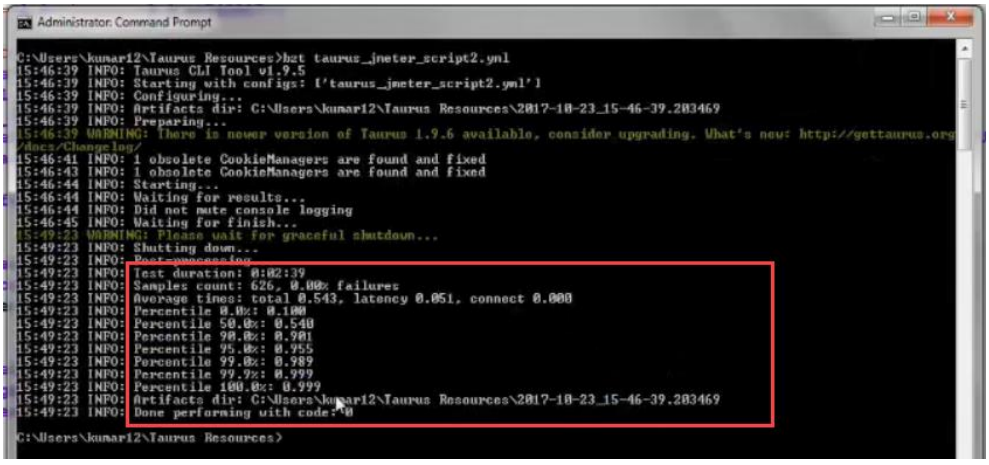
3. Execute the Taurus YAML script from a terminal using the command: **bzt taurus_jmeter_script2.yml.**

4. Press **Enter** to start the script execution. You can see the Taurus console in a few seconds.

5. In the console, you could see the estimated time along with the elapsed time for the two Jmeter scripts (here the ETA is 2 minutes 30 seconds for **jmeter_script1.jmx** as defined in the script), so the test ends after 2 minutes 30 seconds whereas the **jmeter_script2.jmx** execution has started and ended in 6 seconds. You could even observe the active users, latest interval stats, the time taken to connect to the request, latency in the network, and the number of bytes received. It even displays the response code of the website, number of hits on each website that you have entered in the script along with errors per hits.



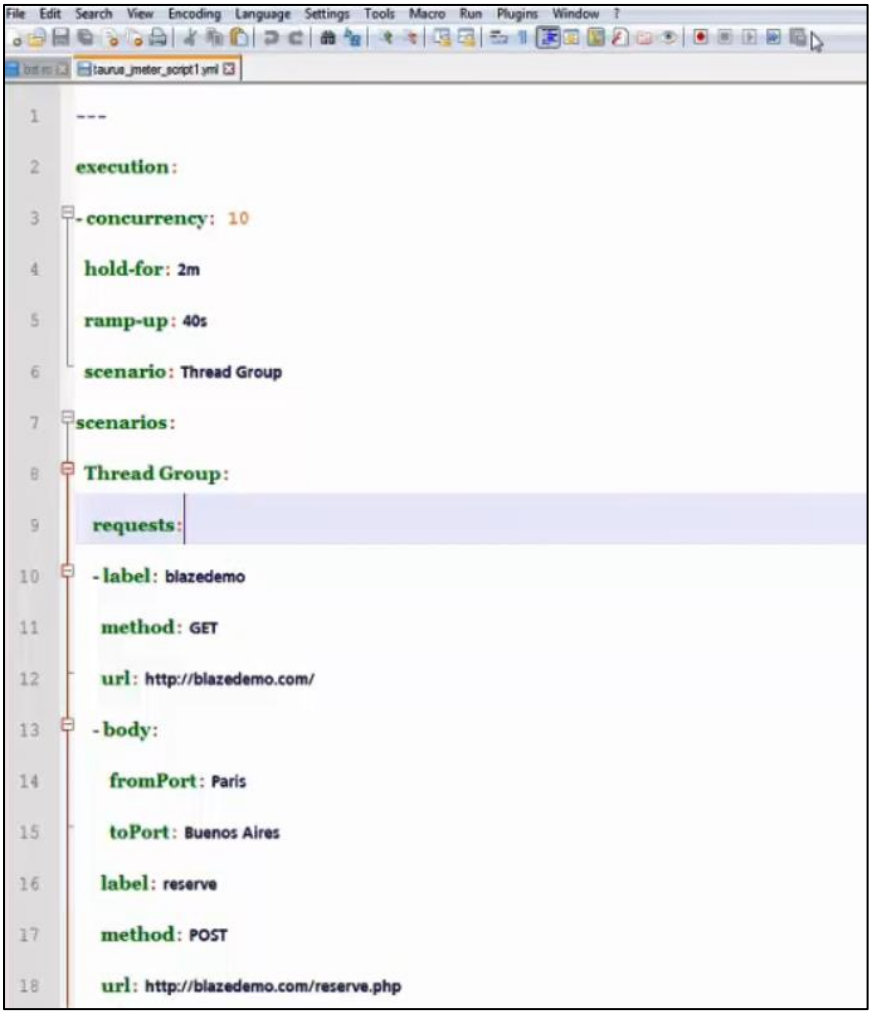
6. This completes the execution of JMeter script through Taurus.



GUI option Usage for JMeter

Here, you will learn how YAML is compiled as JMX and how Taurus uses that JMX file to show up the details or the results of the JMX script on the JMeter GUI.

7. Open the **Taurus_jmeter_script1.yml** from your lab resources folder. The YAML Script used here defines that there will be 10 virtual users and the execution will run for 2 minutes. The ramp-up defines that after 40 seconds all the virtual users will be available for execution.

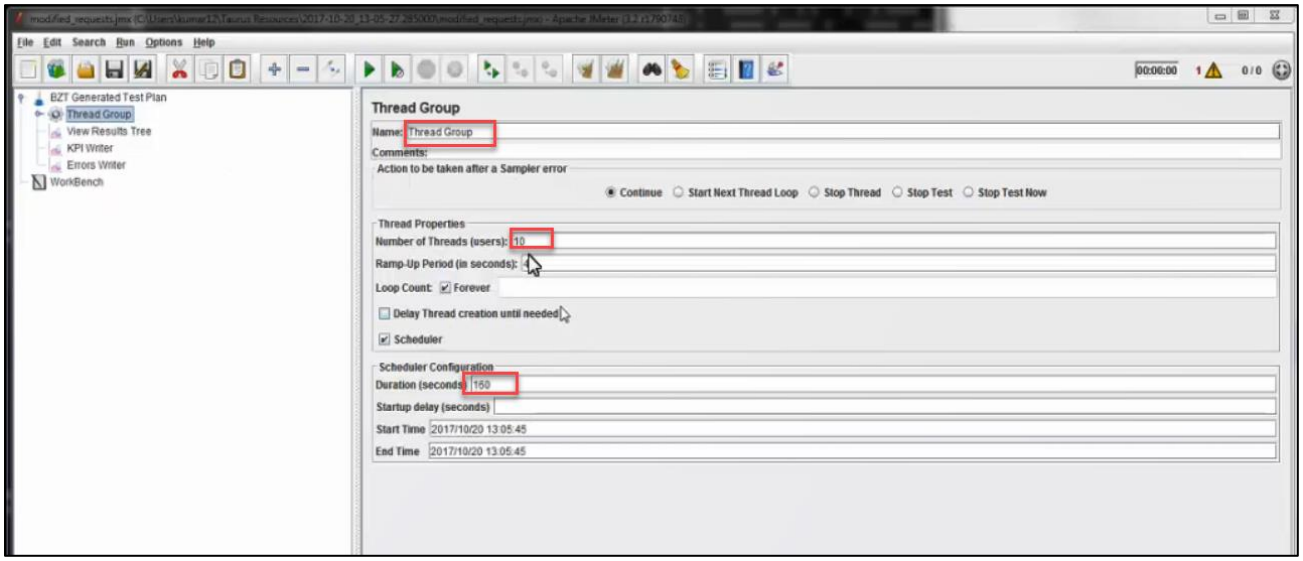


8. Execute the Taurus YAML script from a terminal using the command: **bzt taurus_jmeter_script1.yml -gui**.

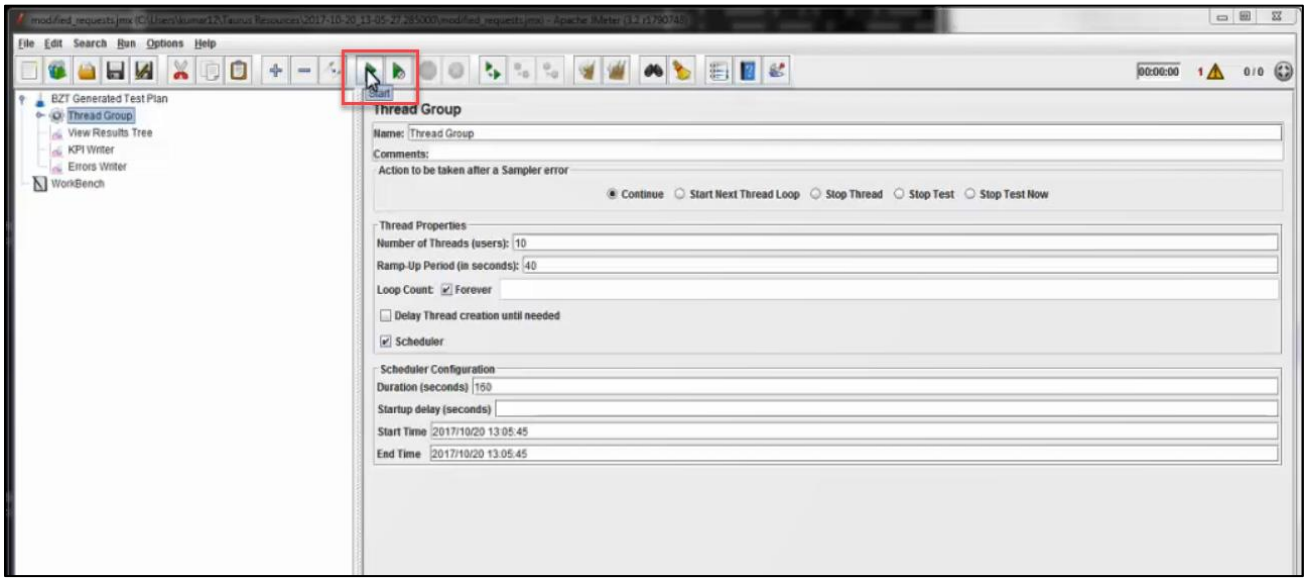
9. Press Enter to start the script execution. You can see the Taurus console in a few seconds along with the JMeter GUI.

   Note: In JMeter GUI, the test script execution does not start automatically.

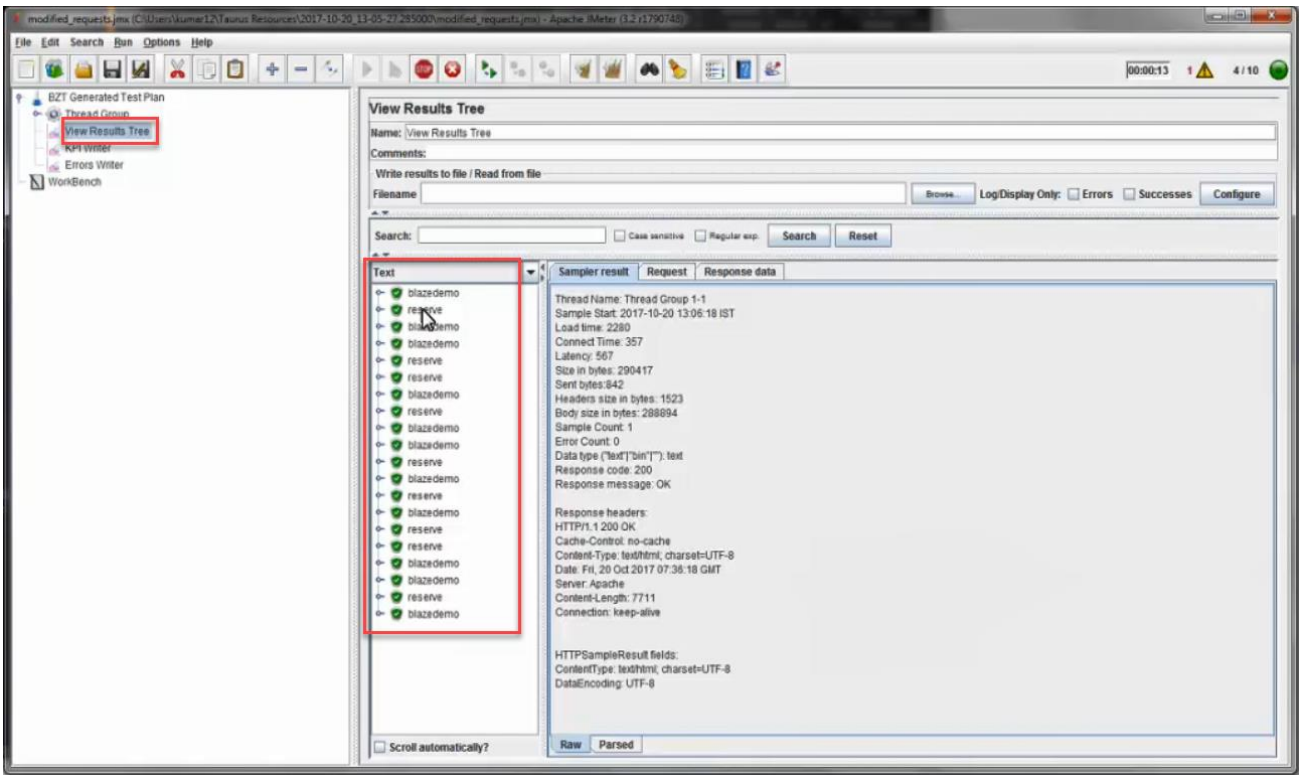
10. From the JMeter GUI, you can observe the number of threads, duration, and ramp-up. The values that you have passed in your YAML script are reflected in the JMeter GUI. You can directly configure the test script from GUI if needed.



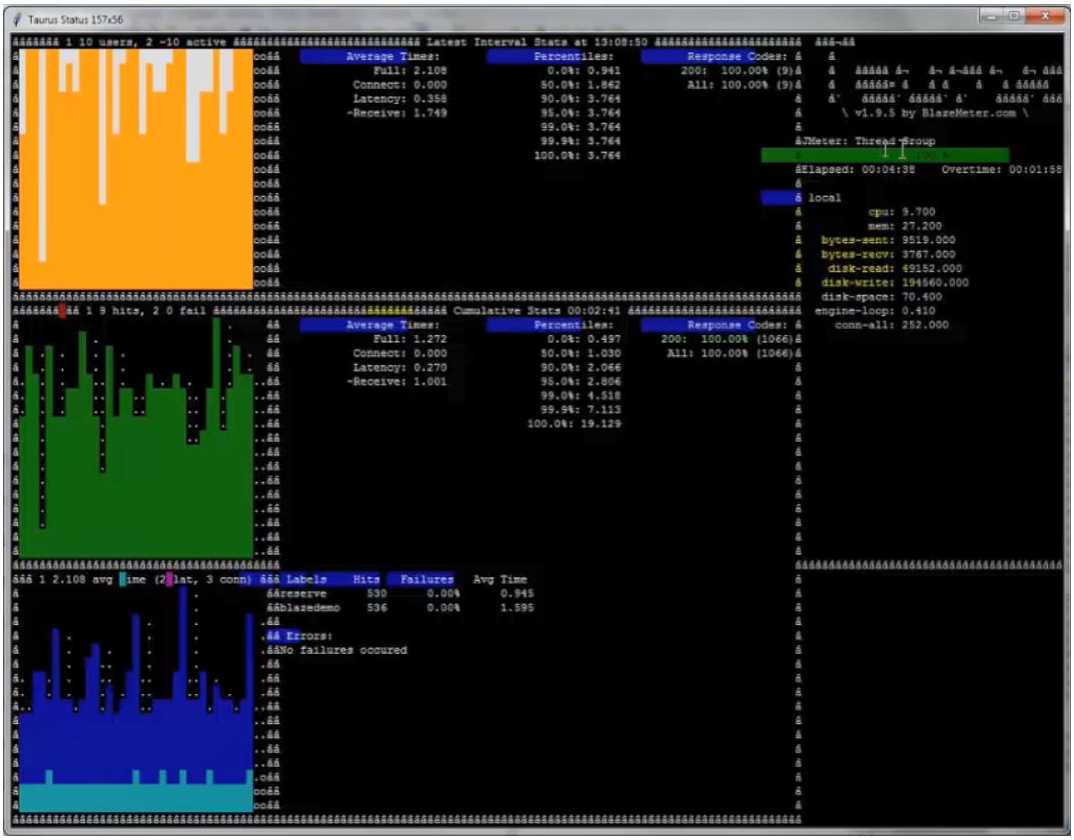
11. Click the **Start** button to start the test execution.



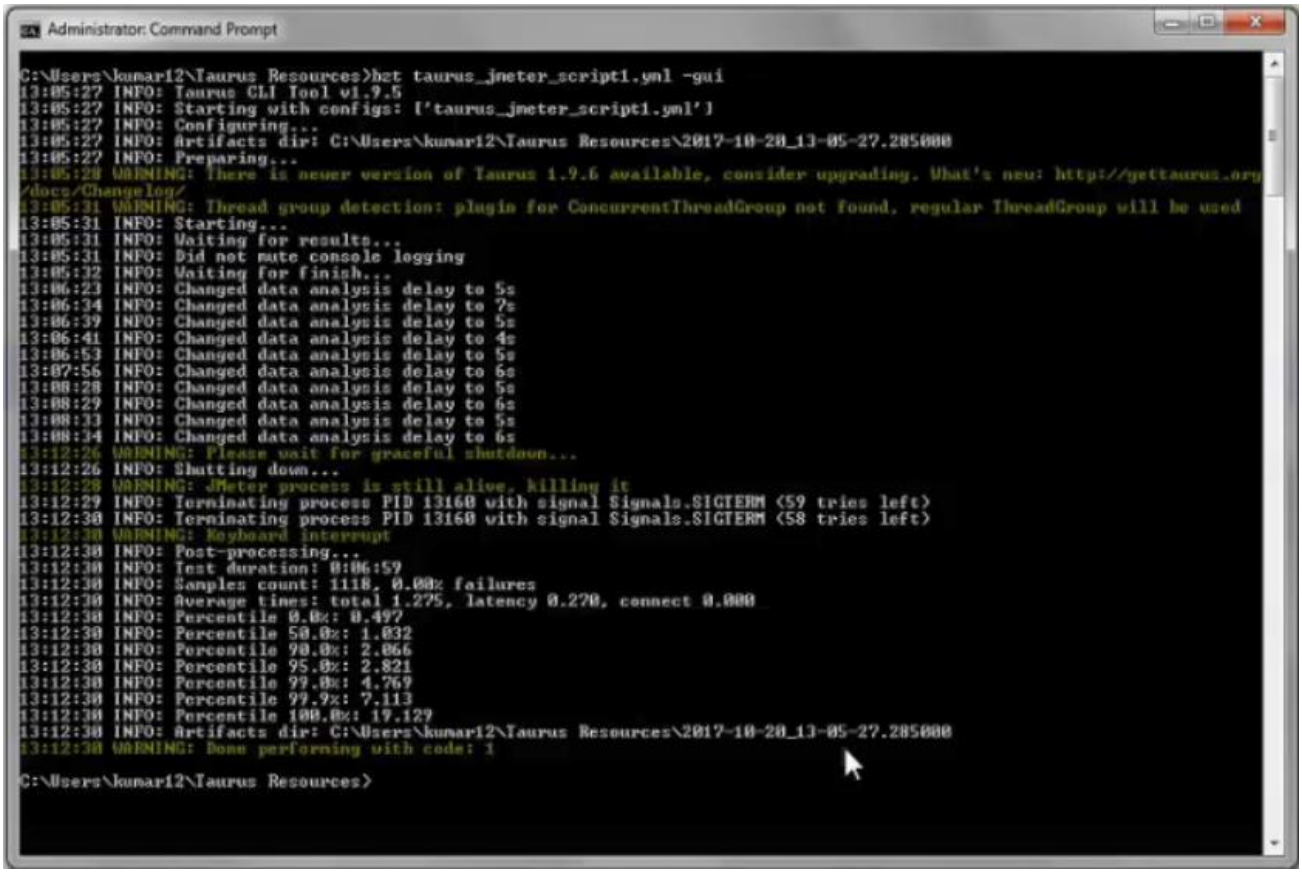
12. The test gets started. Now, click **View Results Tree** to view the execution results. From the View Results Tree, you can observe that URLs that you defined in the script are invoked.



13. You can compare the results from JMeter GUI and start analyzing them from the Taurus Console. This method is best applicable in troubleshooting situations.



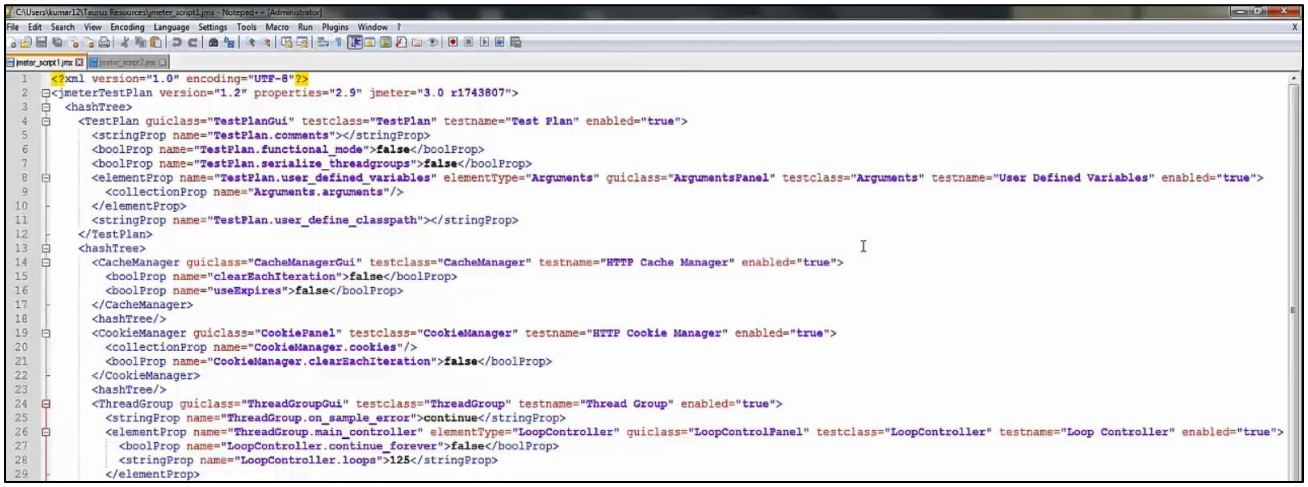
14. Now open the command-prompt and observe that the test execution has successfully shut down.

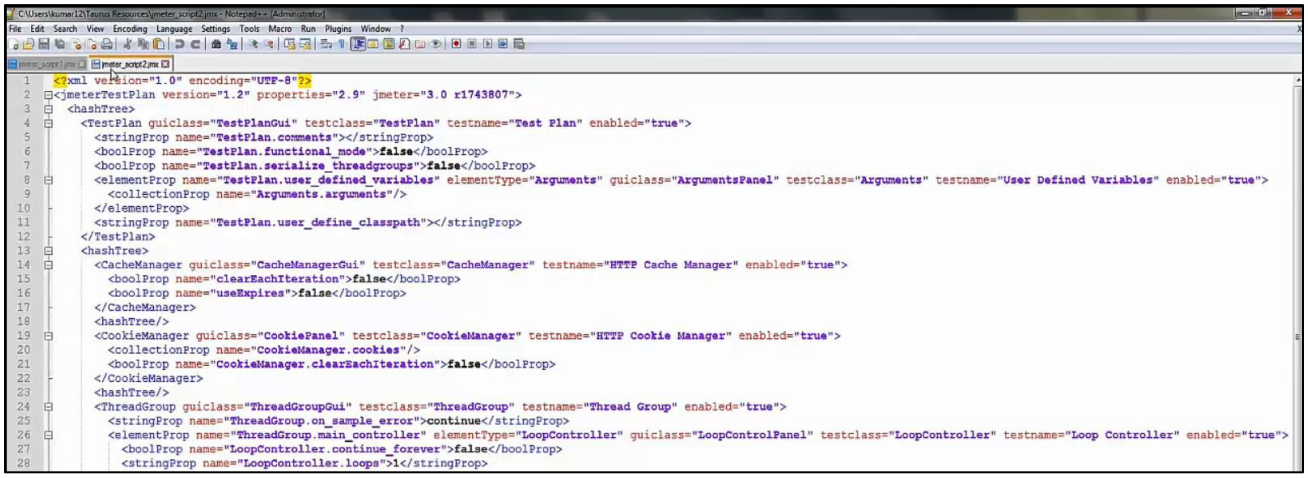


Running Two JMX Scripts Parallelly

Here, you will learn how to run two jmx scripts parallelly.

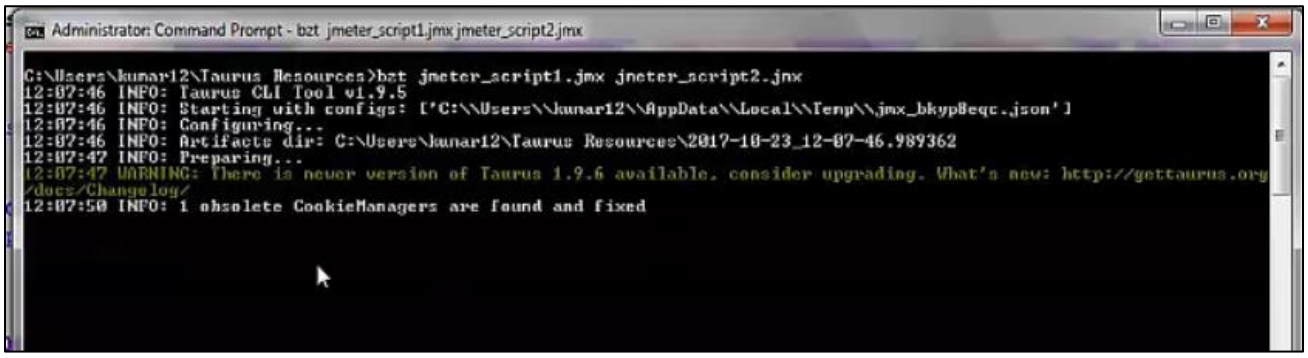
15. Open the **jmeter_script1.jmx** and **jmeter_script2.jmx** from your resources folder.



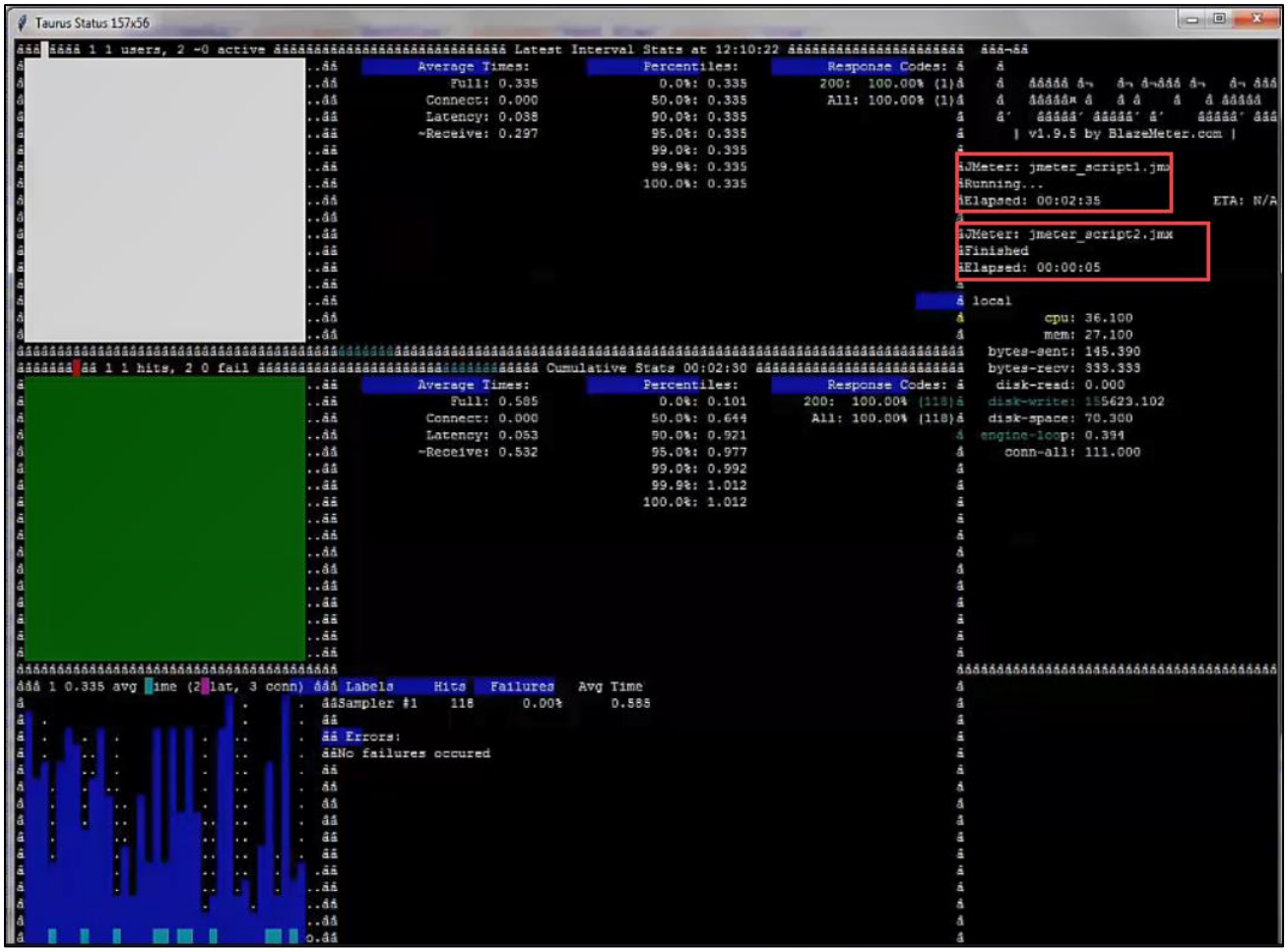


16. Execute the two jmx files from a terminal using the command: **bzt jmeter_script1.jmx jmeter_script2.jmx.**

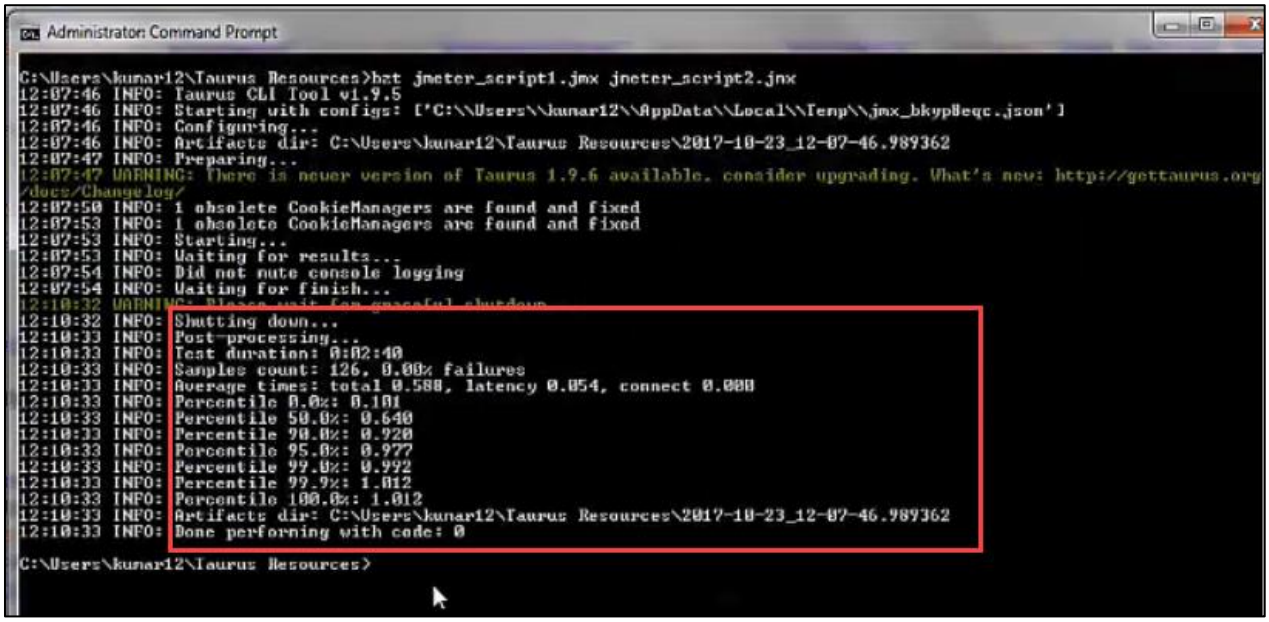
17. Press **Enter** to start the script execution. You can see the Taurus console in a few seconds.



18. Taurus console opens. In the console, you can observe that two jmx scripts are running parallelly. As per the script, the **jmeter_script2.jmx** execution starts and ends in 6 seconds whereas the **jmeter_script1.jmx** is still running as the duration set is 2 minutes 30 seconds. After the successful execution of both the JMX scripts, the Taurus console closes automatically.



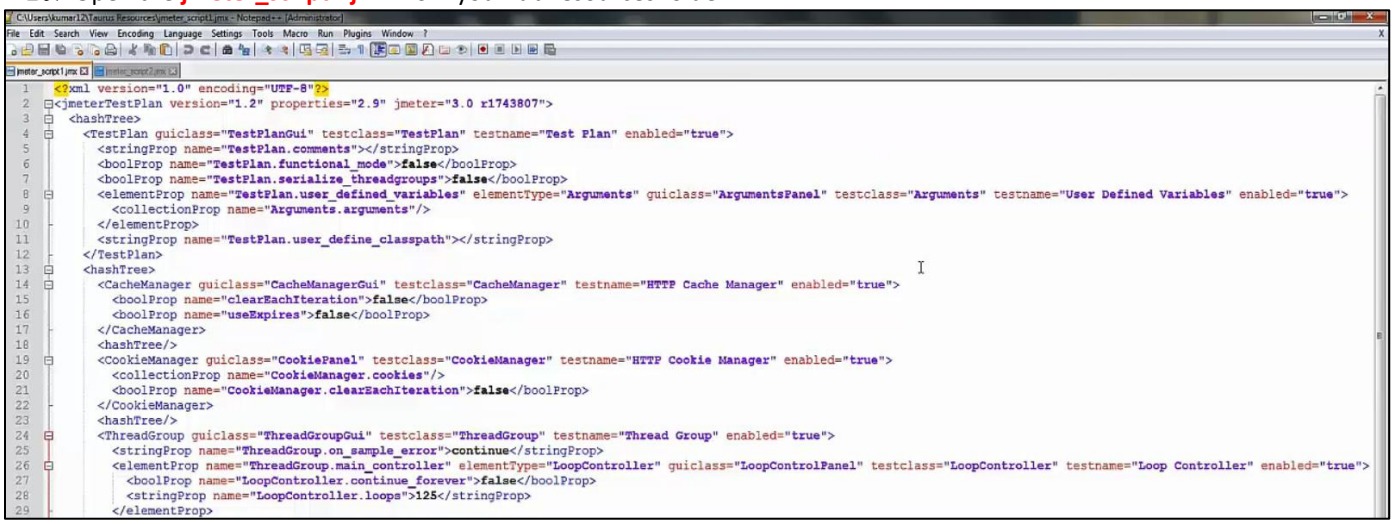
19. Now observe the final statistics of the test execution from the command-prompt. This concludes the demonstration of running jmx scripts parallelly.



Generating a report using -report option

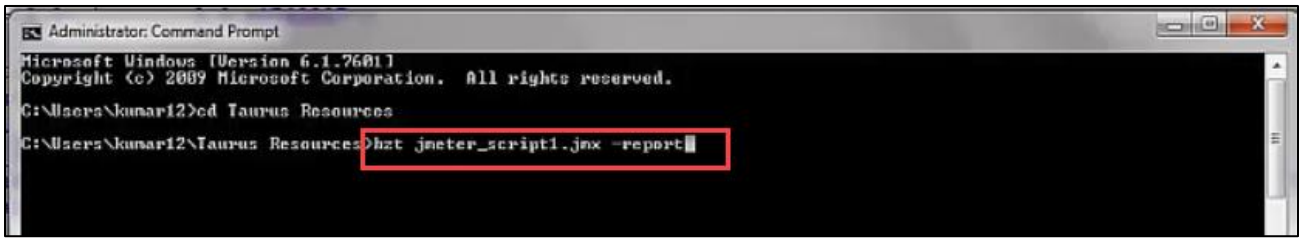
Here, you will learn how to extract a detailed and feature rich report by appending a simple -report option to your jmx file.

20. Open the **jmeter_script1.jmx** from your lab resources folder.



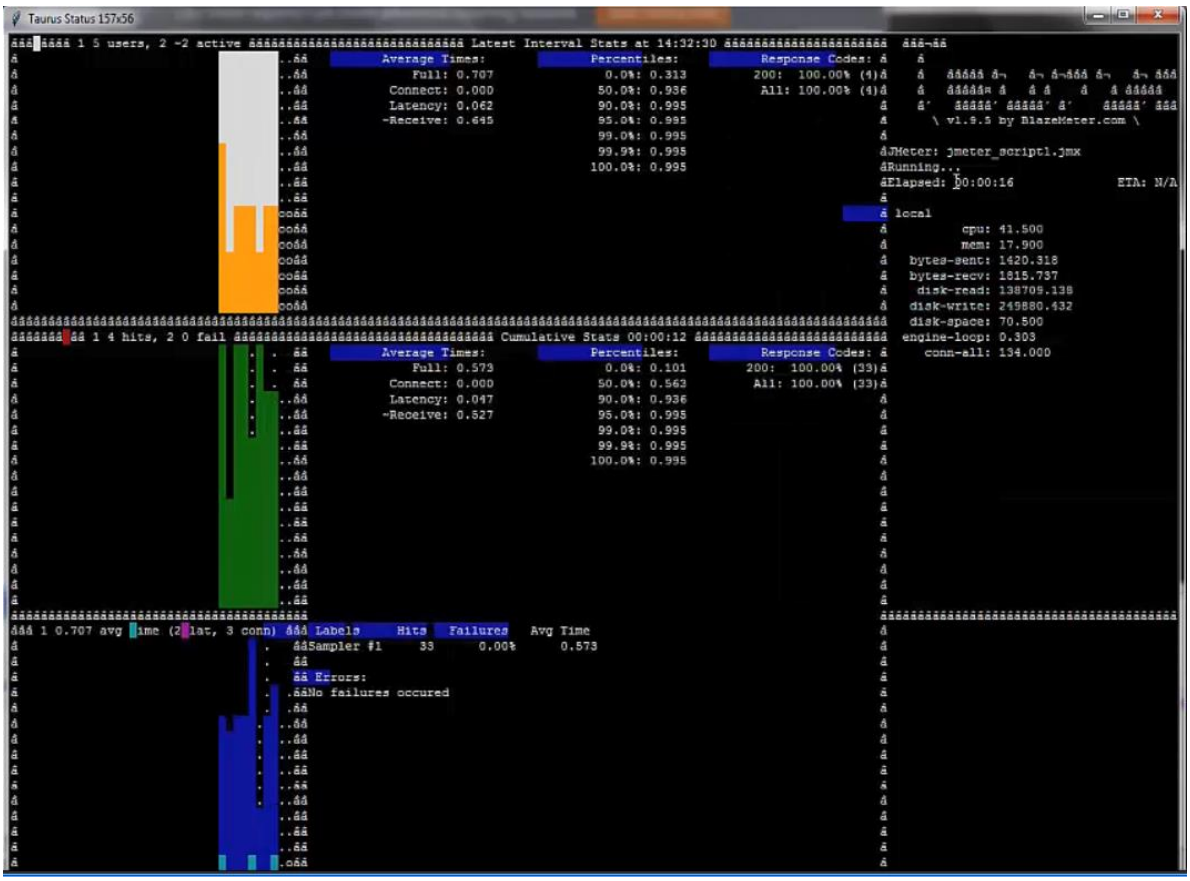
```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="2.9" jmeter="3.0 r1743807">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
      <stringProp name="TestPlan.comments"></stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <CacheManager guiclass="CacheManagerGui" testclass="CacheManager" testname="HTTP Cache Manager" enabled="true">
        <boolProp name="clearEachIteration">false</boolProp>
        <boolProp name="useExpires">false</boolProp>
      </CacheManager>
      <hashTree/>
      <CookieManager guiclass="CookiePanel" testclass="CookieManager" testname="HTTP Cookie Manager" enabled="true">
        <collectionProp name="CookieManager.cookies"/>
        <boolProp name="CookieManager.clearEachIteration">false</boolProp>
      </CookieManager>
      <hashTree/>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group" enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller" elementType="LoopController" guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
          <boolProp name="LoopController.continue_forever">false</boolProp>
          <stringProp name="LoopController.loops">125</stringProp>
        </elementProp>
```

21. Execute the jmx file from a terminal using the command: **bzt jmeter_script1.jmx -report**. The **-report** option will send your results to BlazeMeter's reporting service and you won't need to set anything else up. You will receive the link for your report in the console text, and this will be automatically opened in your default browser.

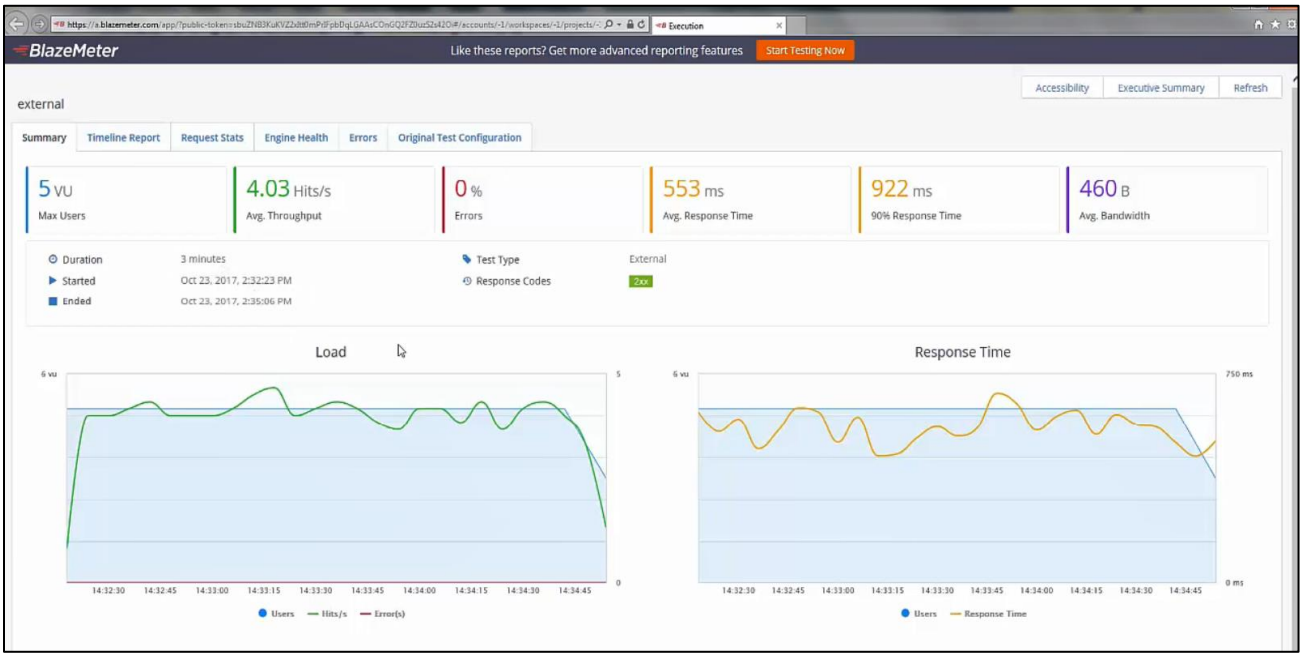


22. Press Enter, the script execution starts. You can see the Taurus console in a few seconds.

23. The test execution starts and the Taurus console is launched. Simultaneously, the BlazeMeter UI opens and starts generating the report. You can observe the JMeter scripts are executed on the Taurus console and review the relevant test information such as response codes, latency, connect time, and average time. The same details can be seen in a BlazeMeter report for better analysis.



24. From the report, you can observe the timestamp for the start and end of the test. You can navigate through different tabs in the BlazeMeter UI to view the results. Unlike Taurus, the report generated on BlazeMeter is permanent.



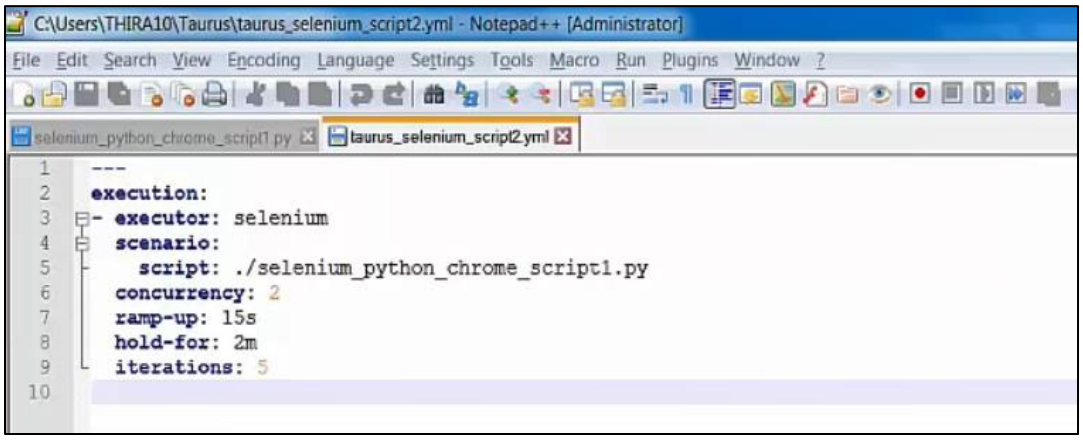
25. This concludes the demonstration.

Lab – Execute a Selenium Script Using Taurus

Goals	Execute a Selenium script using Taurus
Scenario	Availability of a wide range of performance testing tools enables companies to find a solution that can meet any expectation and budget. However, many testing tools are limited when it comes to automating and integrating into the Continuous Integration cycle and using them to manage the functionality workflow might be complicated. From the available testing tools, Selenium is a virtual executor that provides you the ability to run functional tests locally with Selenium WebDriver by choosing an appropriate executor. Taurus automates the execution of native Selenium tests locally and seamlessly switches into the BlazeMeter cloud to run the tests at a massive scale. In this lab, you will use Taurus to execute the test suite in a loop until the desired number of iterations are complete or the hold-for time is exceeded. In this demonstration, you will: ▪ Write a simple YAML and Selenium script (written in Python language) ▪ Execute the script ▪ View results in the Taurus console
Time	10 minutes

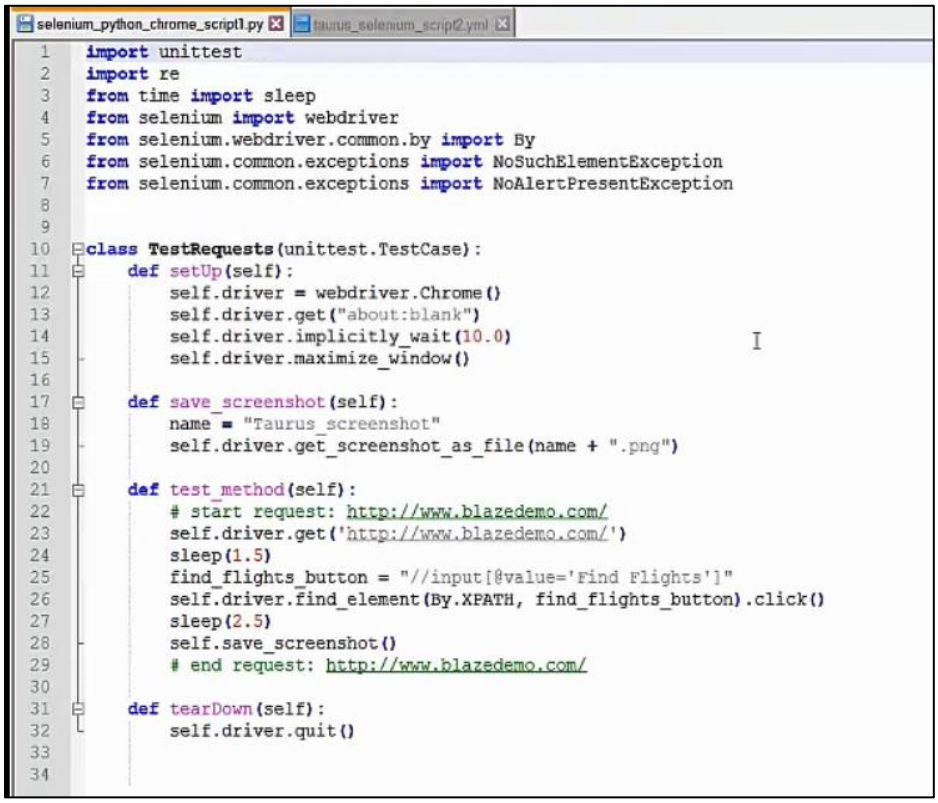
Instructions:

1. Open the **Taurus_selenium_script2.yml** file from the lab resources folder saved on your local system. This selenium script is written in the Python language.



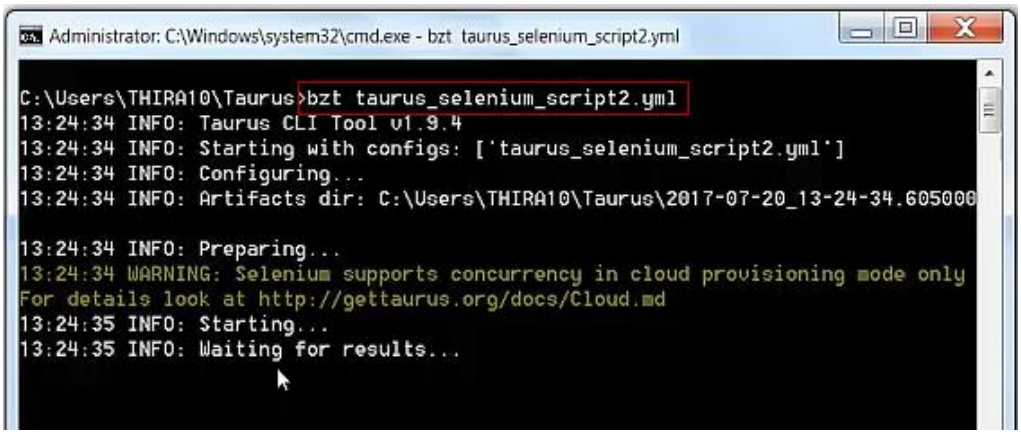
2. Open the python script **selenium_python_chrome_script1.py** saved on your local system using Notepad++ and review the contents of the file. The script is written in Python language.

3. As per the script, the web driver is Chrome. The first request in the script is to open **about blank** page and it waits for 10 seconds, then maximizes the window, and takes the screenshot of the web page. The test invokes the Blazedemo.com, reserves the flights after a delay of 1.5 seconds, and then saves the screenshot.



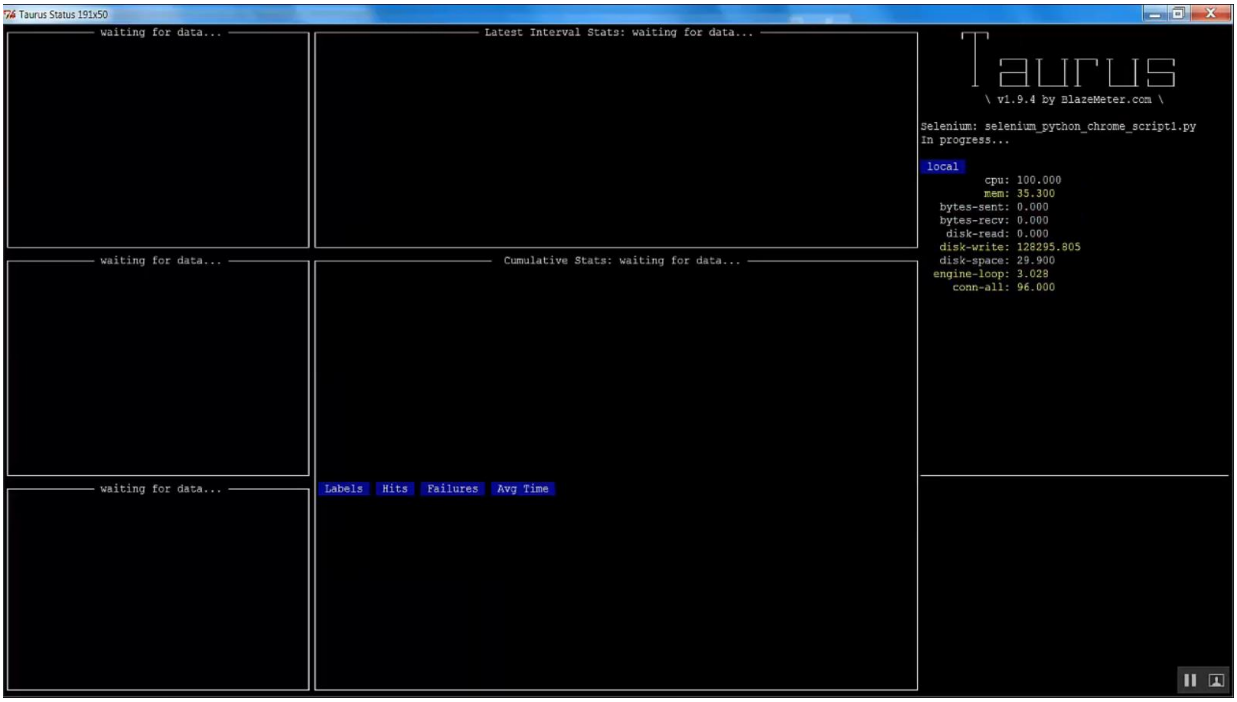
4. Open a Terminal window.

5. Run the Selenium script from a terminal using the command: **bzt taurus_selenium_script2.yml.**

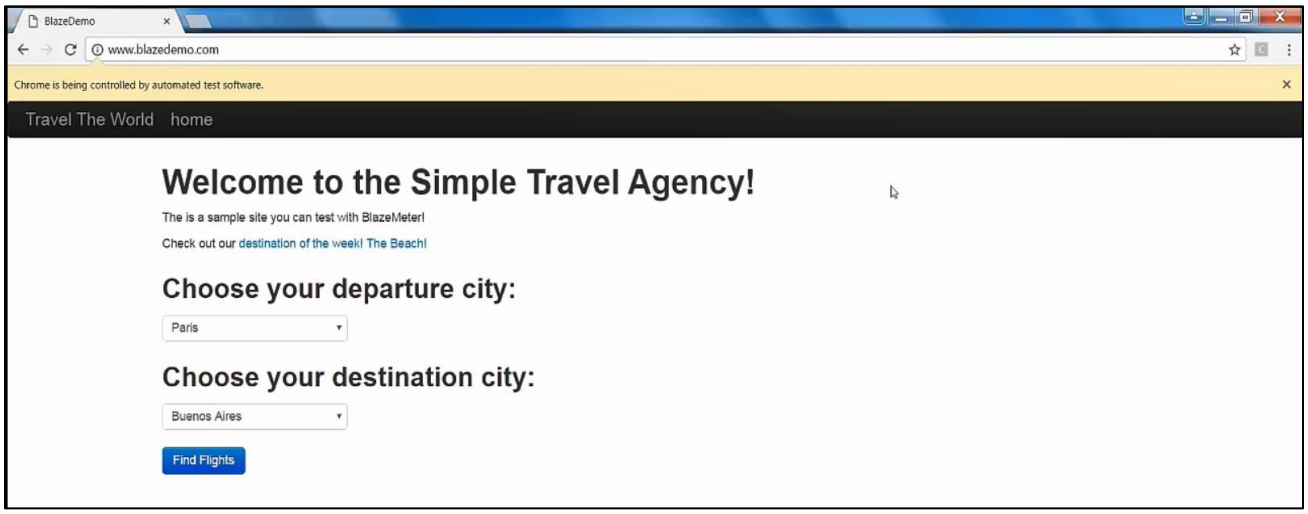


6. Press **Enter**, the execution starts, and you can see the Taurus console.

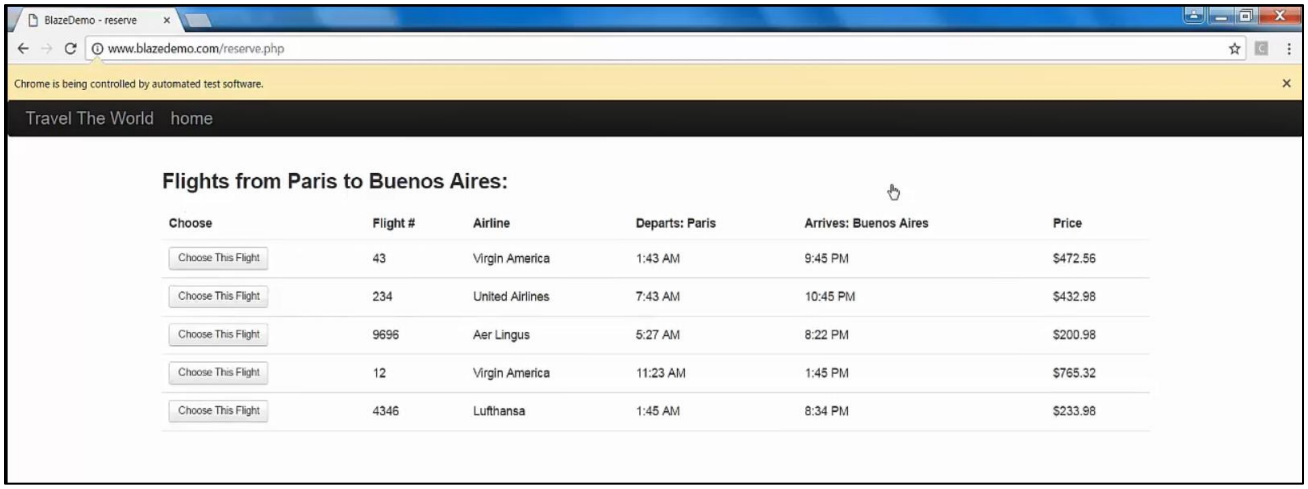
7. Note the Selenium Executor on the top right executing the selenium script. Review the results in the console.



8. From the console, you could observe that the web browser starts, and opens the about blank page. The web page maximizes as defined in the script. Later, it captures the screenshot of the **www.blazedemo.com** as per the execution sequence in the Python script.

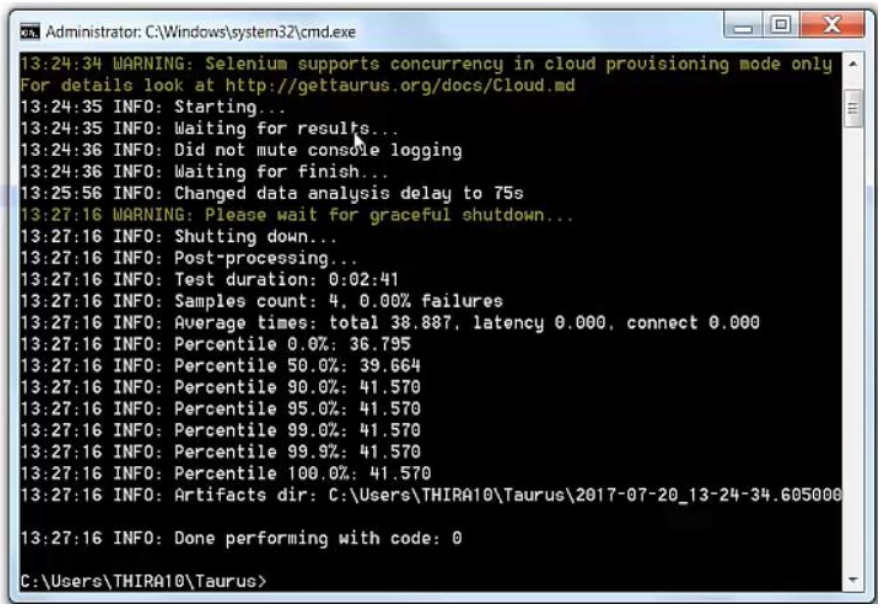


9. The test execution opens the **blazdemo.com/reserve.php** web page and reserves the flights.



10. As per the python script, the script execution starts again after a delay of few seconds and the console gets closed.

11. This completes the execution of selenium script.



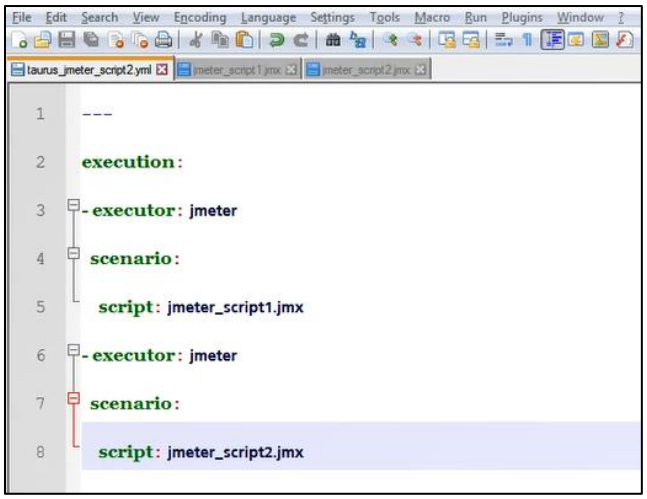
Lab – Execute Taurus Script with BlazeMeter Report

Goals	Execute a Taurus script with BlazeMeter report.
Scenario	The greatest advantage of running a JMeter script using Taurus is that you get access to the BlazeMeter's reporting service. While many of the testing tools focus on execution and less on reporting, the BlazeMeter reporting service allows you to access test results conveniently and interactively. You can use the reports to compare different test executions, and monitor trends over time. This is available on all BlazeMeter pricing tiers including the free version, and it even works in the anonymous mode. In this demonstration, you will: ▪ Run the Taurus script ▪ Review the results in Taurus console ▪ Review the results in BlazeMeter report
Time	10 minutes

Instructions:

1. On executing a Taurus script, the console opens and shuts down after the time limit as specified in the script. To have a thorough review of the results generated, the results console must be available for a long time. So, instead of viewing the results on Taurus Console, you can see the test execution results in BlazeMeter and generate a permanent report for future reference.

2. Using Notepad++, open the file **taurus_jmeter_script2.yml** from the lab resources folder that you have saved on your local system.

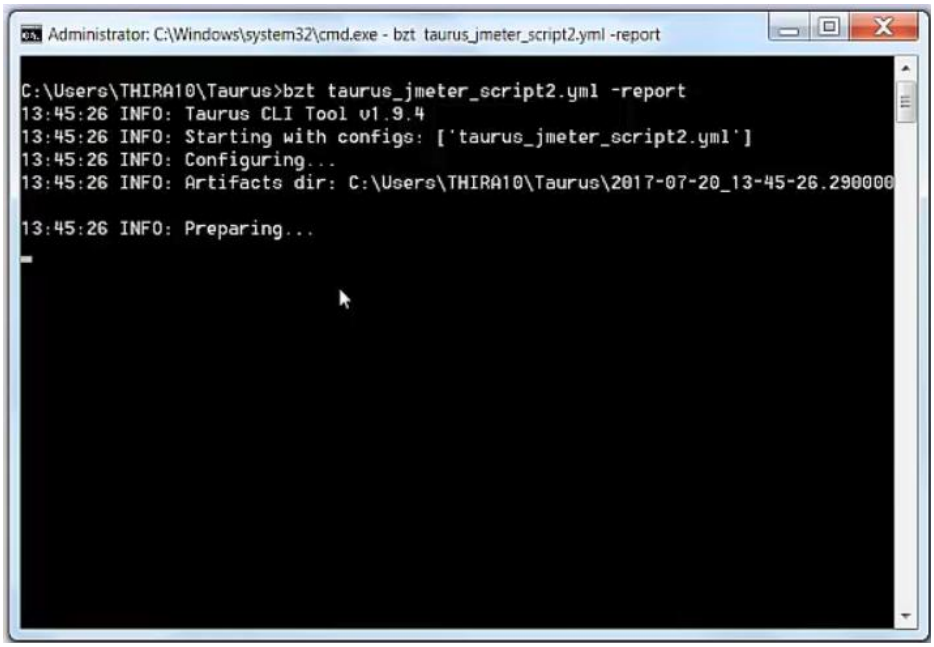


3. Enter the command **bzt taurus_jmeter_script2.yml -report** to run the Taurus YAML script from the terminal.

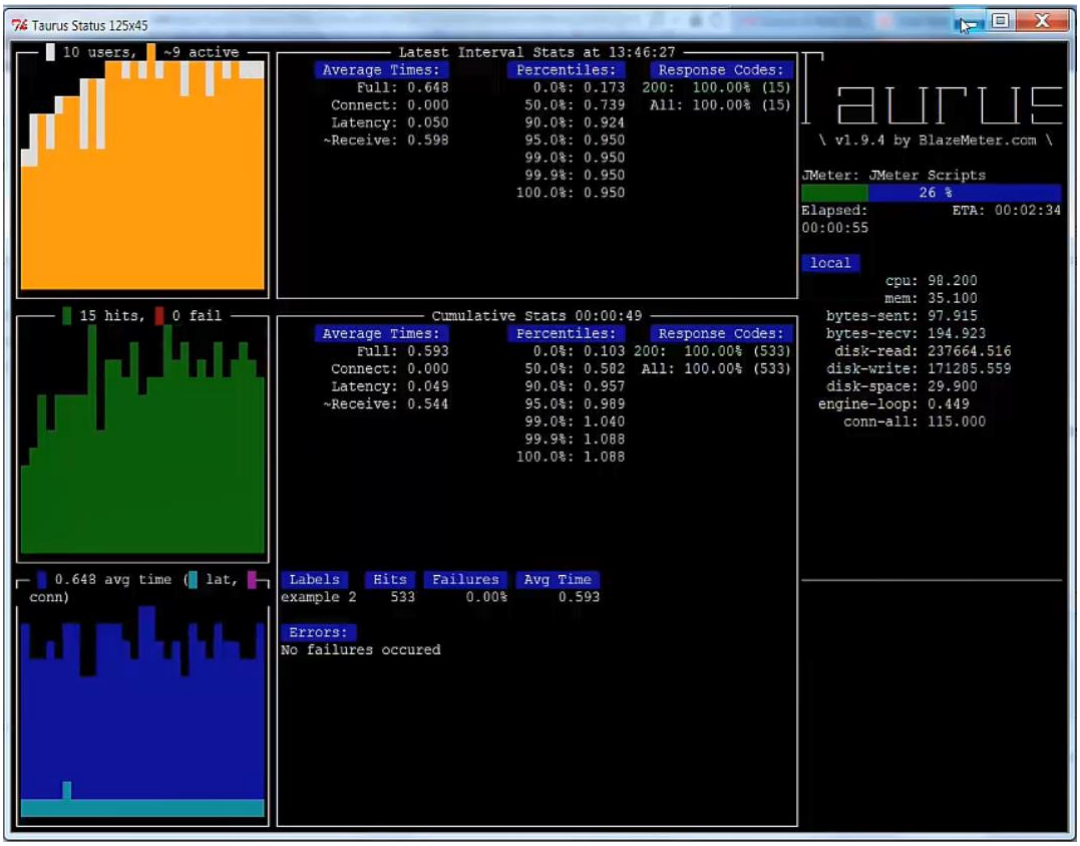
Note:

- You can give any Taurus script file name for which you want to generate the report in BlazeMeter.
- The **-report** option will display BlazeMeter reports for the Taurus test without using any BlazeMeter account. To use a specific BlazeMeter account for the reports, you need to use BlazeMeter API Key.
- The -report command line switch will send your results to BlazeMeter's reporting service and you do not need to set anything. You will receive the link for your report in the console text, and this will be automatically opened in your default browser.

4. Press **Enter** to start the execution.

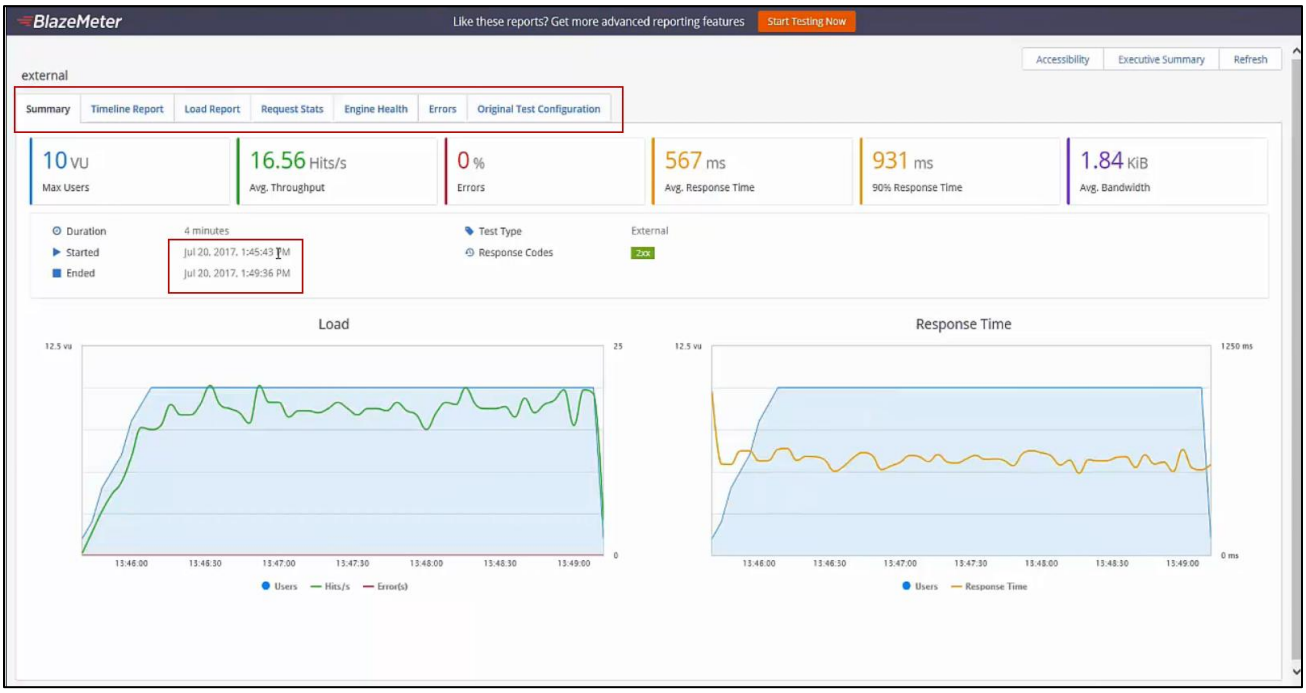


5. The test execution starts and the Taurus console is launched. Simultaneously, the Blazemeter UI opens and starts generating the report. You can observe the JMeter scripts are executed on the Taurus console and review the relevant test information such as response codes, latency, connect time, and average time. The same details can be seen in a BlazeMeter report for better analysis.



6. From the report, you can observe the timestamp for the start and end of the test. You can navigate through different tabs in the BlazeMeter UI to view the results. Unlike Taurus, the report generated on BlazeMeter is permanent.

Report Type	**Description**
Summary Report	Provides main dashboard view of your test while it is running and after it has finished. It appears as soon as your test starts to collect data.
Timeline Report	Enables you to view many different types of KPIs within one graph.
Load Report	Displays values for all requests made during the test and an individual row for each named request in your test.
Engine Health Report	Displays the performance indicators received during the test while monitoring JMeter console(s) and engine(s).
Errors report	Contains the errors that were received by the web-server under the test because of HTTP request.
Original Test Configuration Report	Displays the complete load configuration properties along with session ID. This ID will be used as a parameter in the API to identify a specific session of your BlazeMeter test.



7. This concludes the execution of Taurus script on BlazeMeter report.

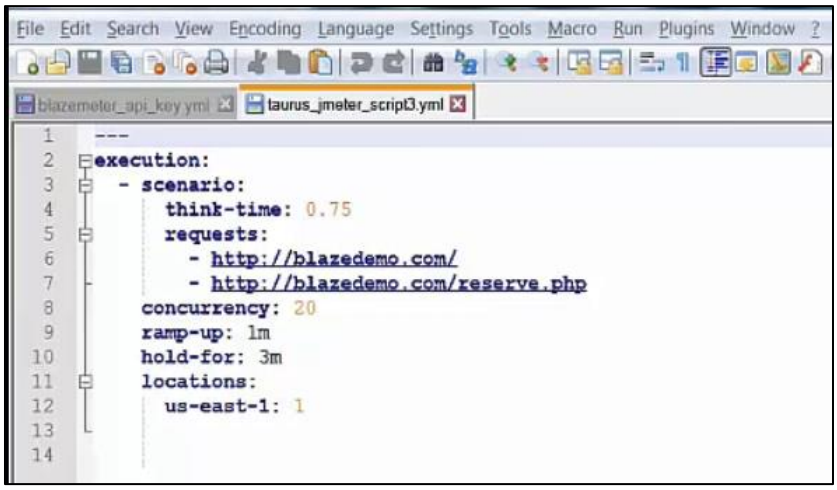
Lab – Execute a Taurus Script by Using a BlazeMeter Account

Goals	Execute a Taurus script by using a BlazeMeter account.
Scenario	Running JMeter scripts from your local system is not a scalable option because you are limited to the use of resources available on your local computer. However, Taurus provides you an option of cloud provisioning by using BlazeMeter. You can use cloud provisioning, to run your JMX scripts on the cloud by accessing your BlazeMeter account. To run this lab, you don't need to be a premium customer of BlazeMeter since a free tier account allows you to execute cloud tests. You can choose one of the various cloud locations provided by BlazeMeter to generate the load. In this demonstration, You will: ▪ Create a JMeter YAML script ▪ Access the BlazeMeter application ▪ Identify API key ▪ Place the API key in .bzt-rc file ▪ Execute the script ▪ View results in Taurus console ▪ View the results in BlazeMeter UI ▪ Generate Executive summary report
Time	10 minutes

Instructions:

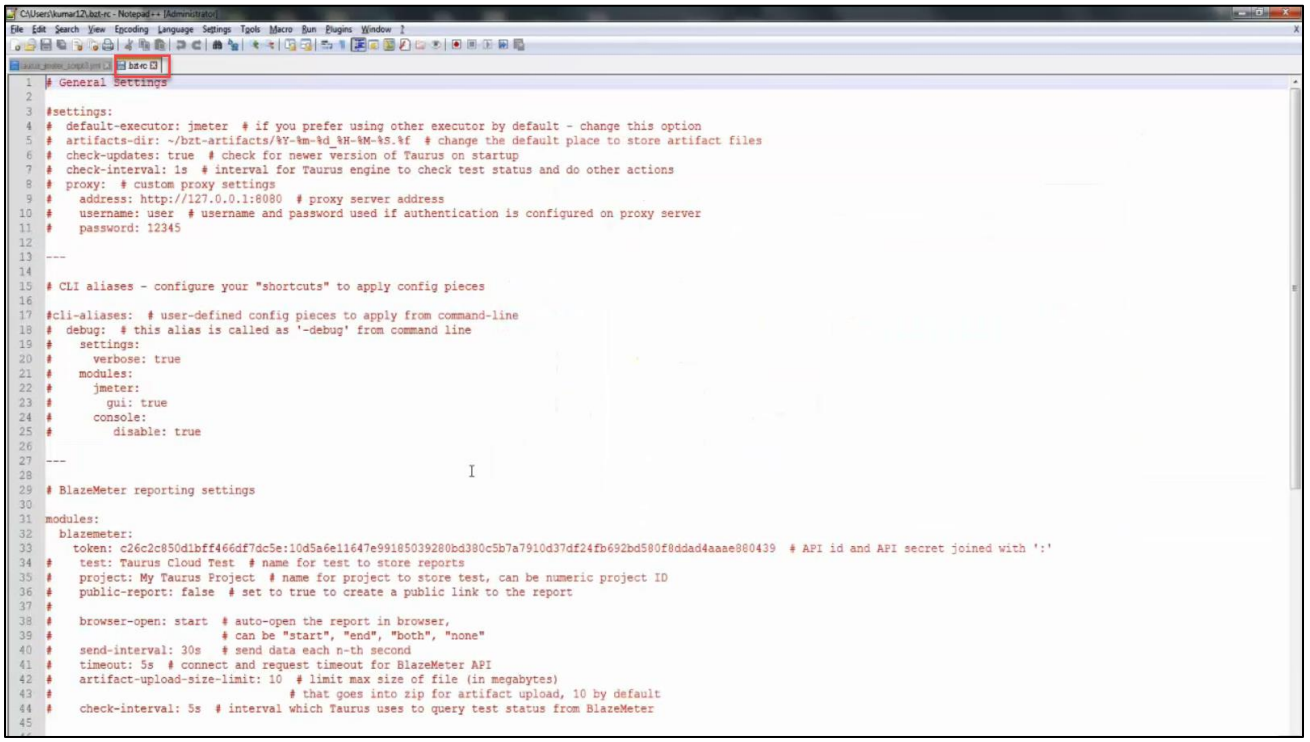
1. Open the **taurus_jmeter_script3.yml** from the lab resources folder on your local system. This is the JMeter script written in YAML. In this lab, you will send the requests to Blazedemo.com and Blazedemo.com/reserve.php and generate the load from us-east-1 cloud location.

   Note: If no location is specified in the script, the Taurus will utilize the default Google location.



2. Now, you will specify the API key for cloud provisioning in the .bzt-rc file. It is recommended to place the token setting in your personal per-user config ~/.bzt-rc file to prevent it from being logged and collected in artifacts.

3. Navigate to your home directory and open the .bzt-rc file from the folder where it is located.



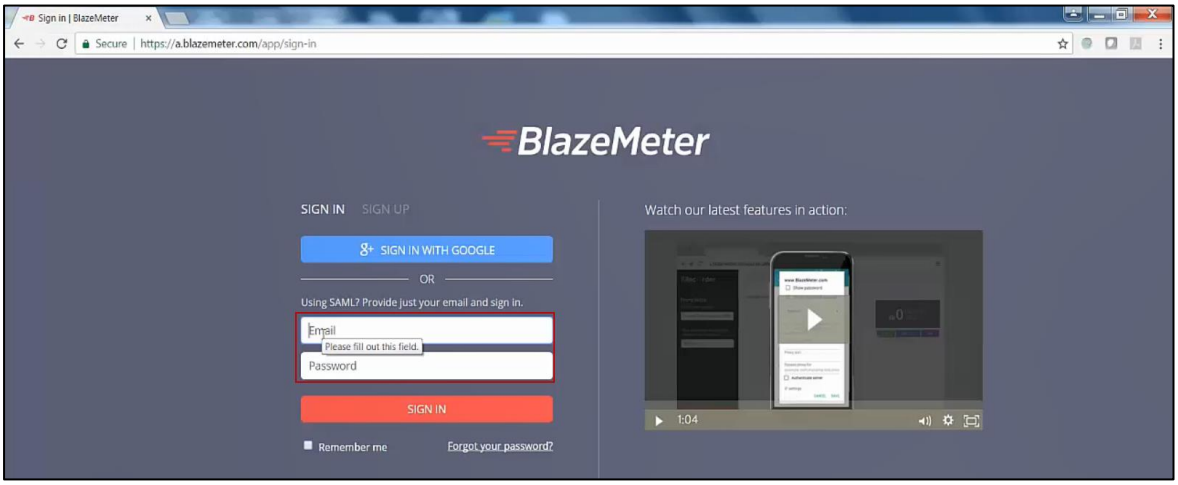
4. To specify the API key, you must scroll down to the BlazeMeter reporting settings in your .bzt-rc configuration file and search for token keyword. The key ID and secret are joined with single colon ':'.

5. To enter the API key, you must access the BlazeMeter application from your browser.

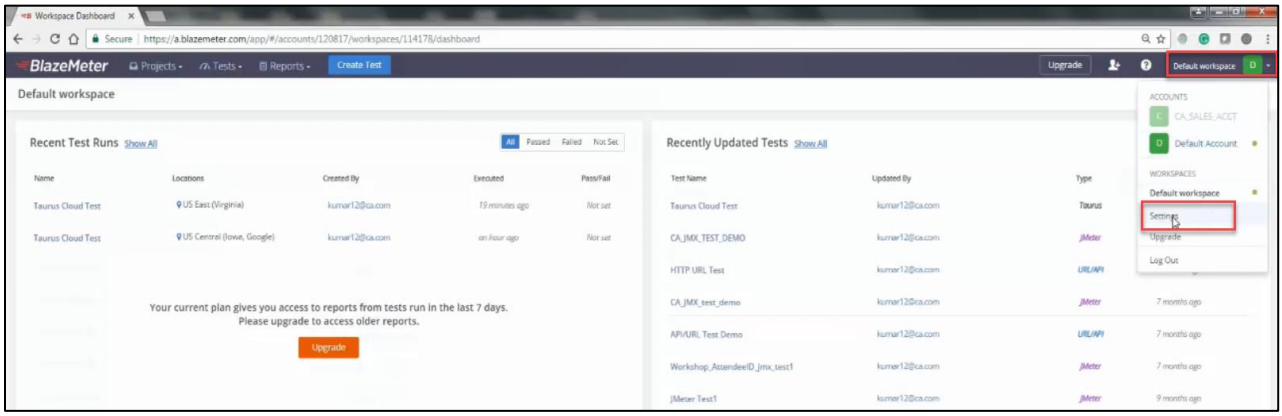
6. Enter the URL https://a.blazemeter.com/app/sign-in in a browser.

7. At the login prompt, enter your login credentials and click **SIGN IN**.

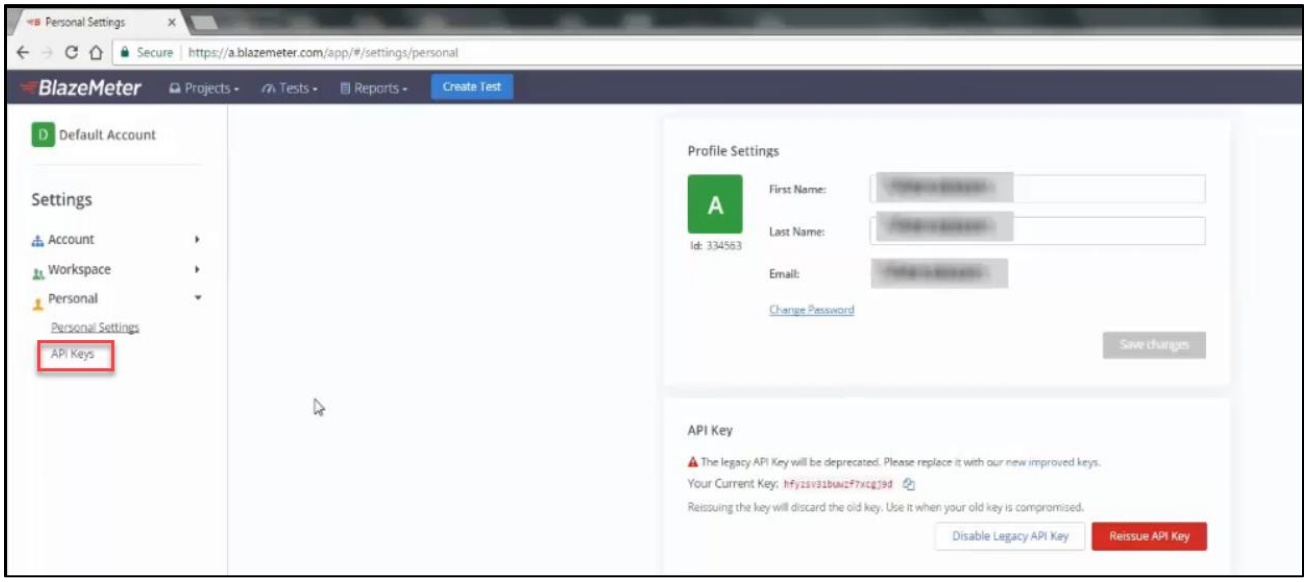
Field	Value
User Name	**Enter your registered username**
Password	**Enter a valid password**



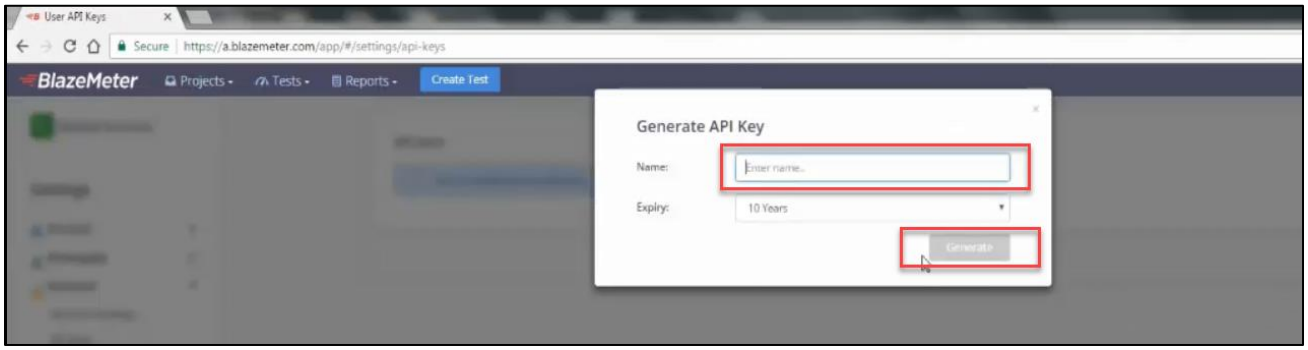
8. On successful login, the BlazeMeter application opens. From the BlazeMeter UI, you could see the **'User Account'** at the top right corner of the screen. Click the drop-down icon and then click **Settings**.



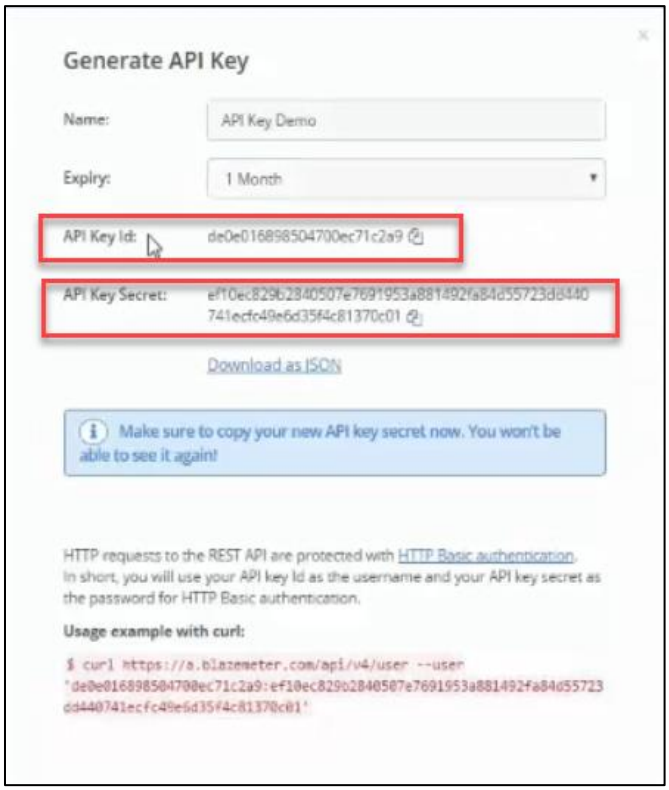
9. Now, click the **'API Keys'** option under Settings.



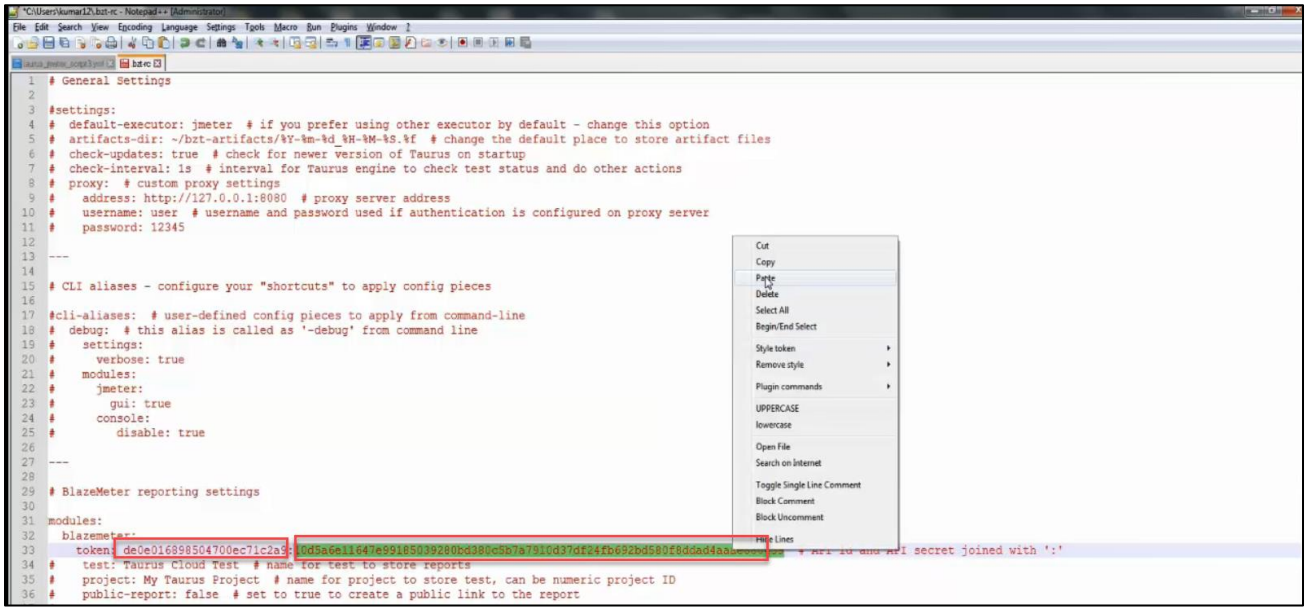
10. You can observe that no API keys are generated for your account. Now, click the '**+**' symbol to generate a new API key. Enter a name for the API and key and select the expiry time for your API key from the drop-down.



11. Now, the API token is generated. Observe the API key Id and its secret. These two values must be placed in the .bzt-rc file. Copy the **API Key Id** value and its **secret key** form using the copy to clipboard icon.

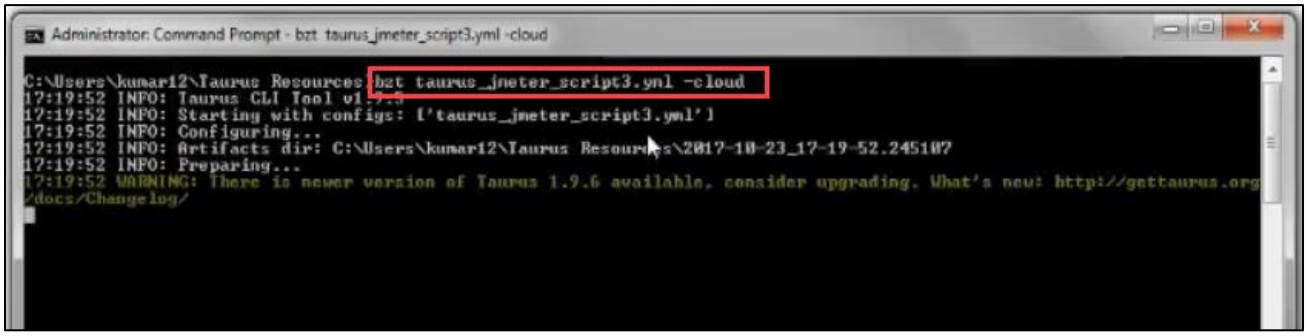


12. Return to the .bzt-rc file and paste the API key and the API Key Secret that you have copied as a value for the token element. Click **Save** to save the script.

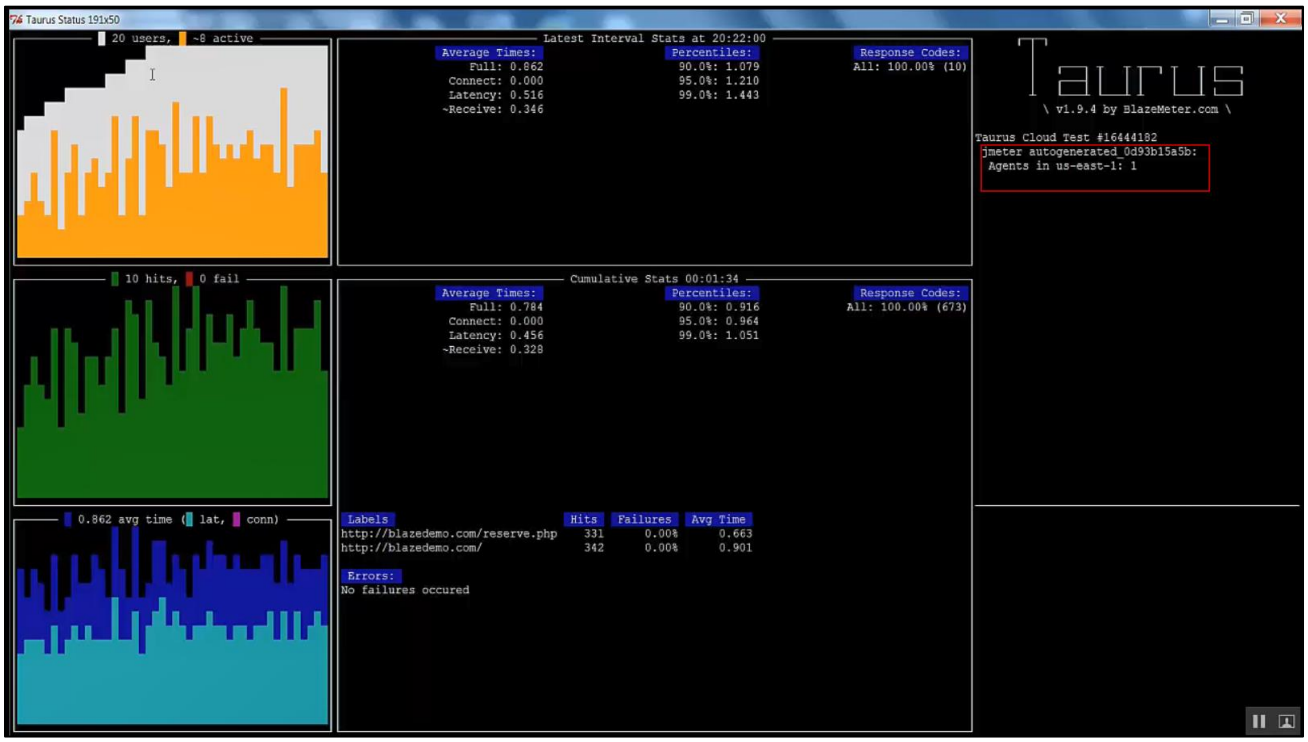


13. Run the Taurus script YAML from a Terminal using **bzt taurus_jmeter_script3.yml -cloud** to generate the load from the cloud.

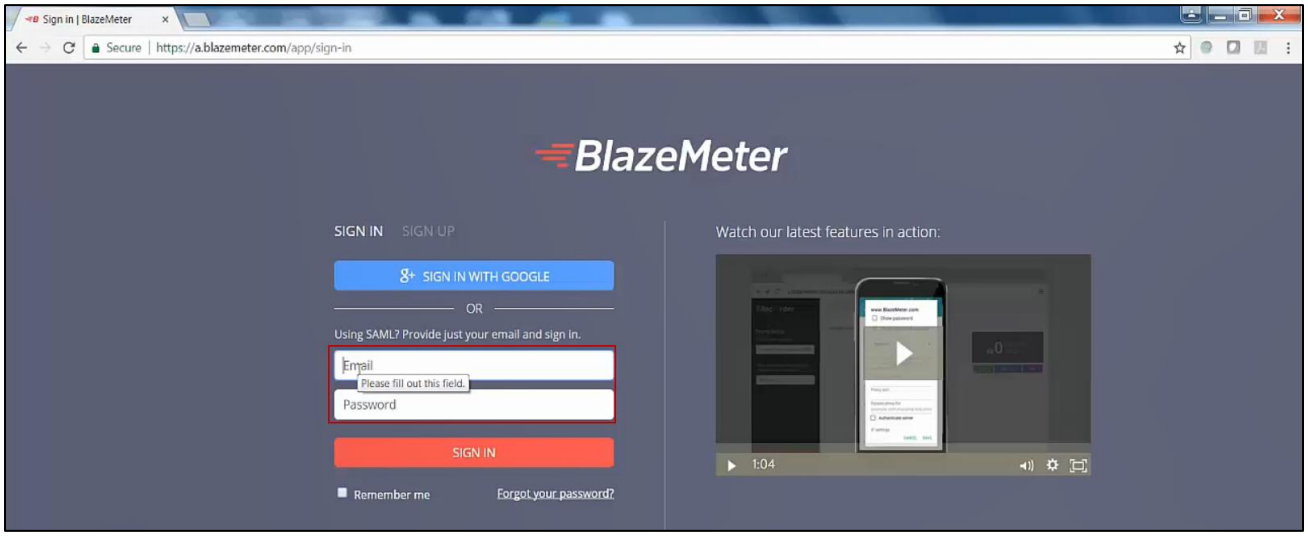
14. Press **Enter**, and the execution starts. You could observe that configuration files are getting created using the YAML Script.



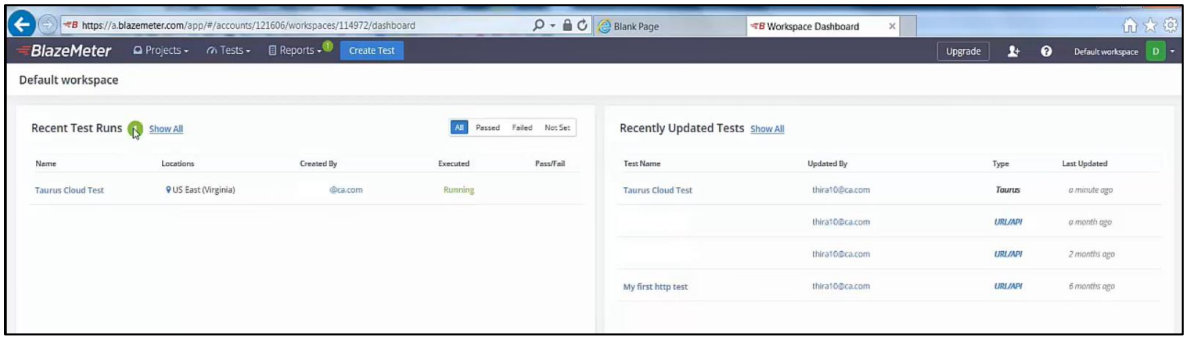
15. Here you could observe that execution is using **us-east-1** as the agent to run the test.



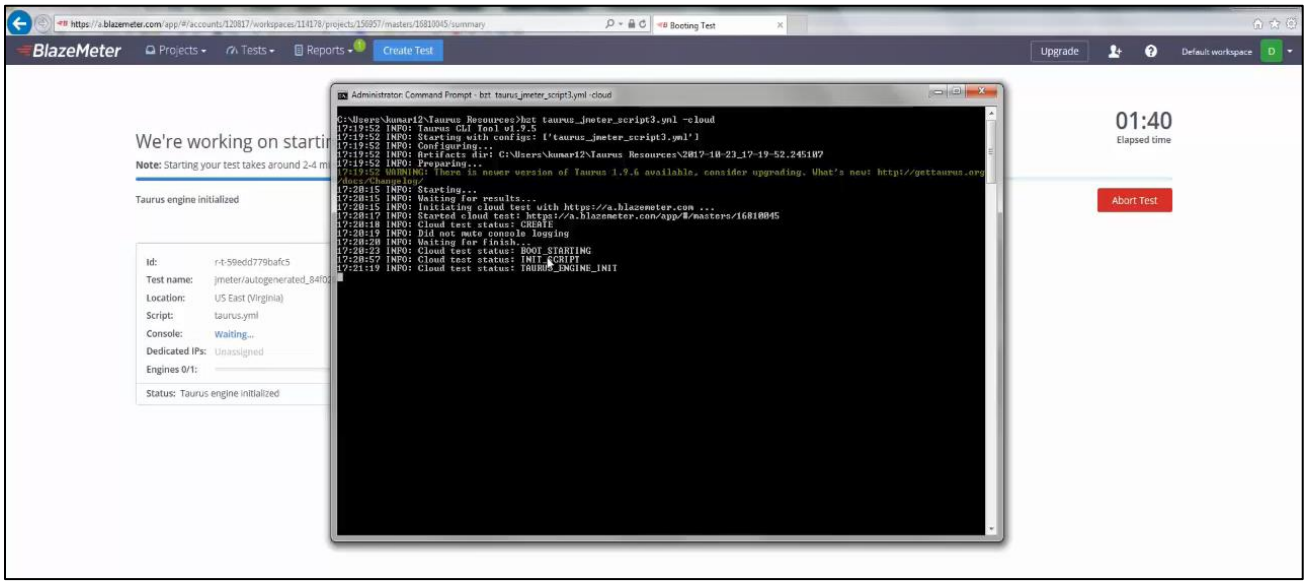
16. A new BlazeMeter browser session starts. Now, access the Blazemeter application by providing valid login credentials.



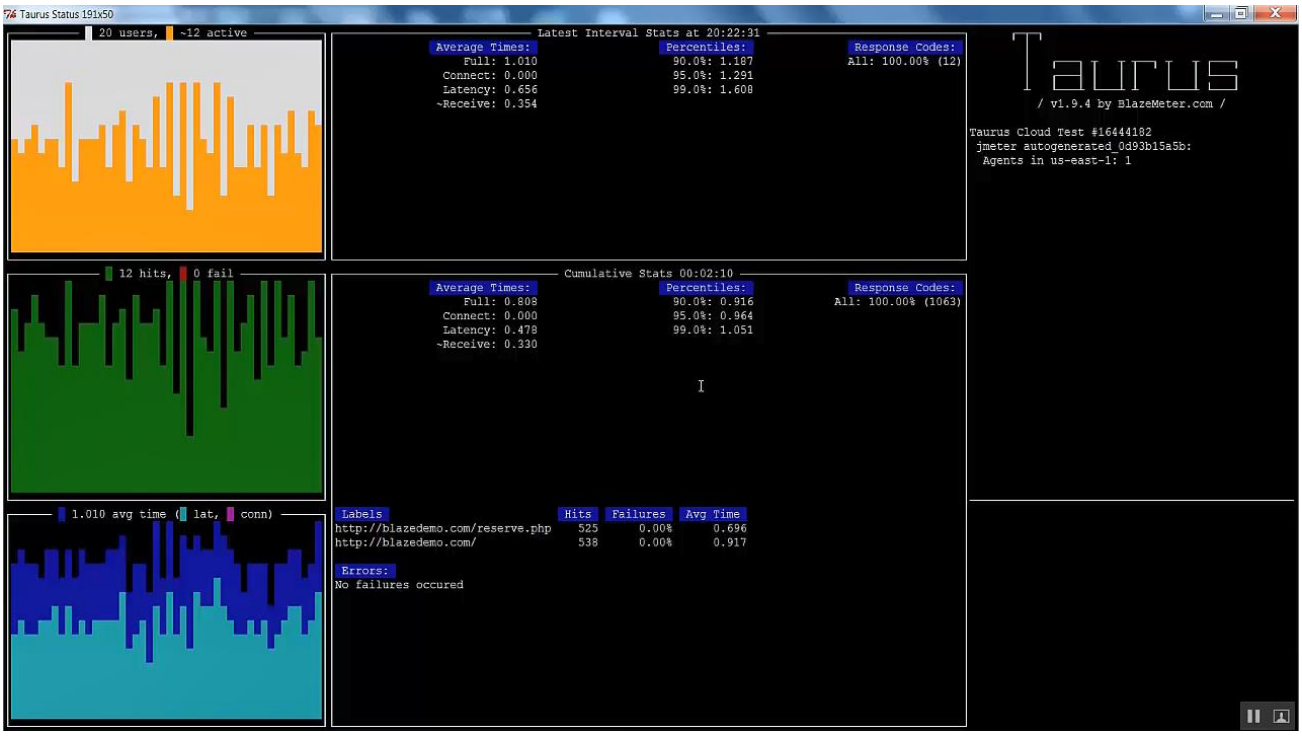
17. The BlazeMeter application opens, and you can see that Taurus Cloud Test has started. Click the **Taurus Cloud Test.**



18. You could observe that the engines are starting and the execution status can be seen in the command prompt.

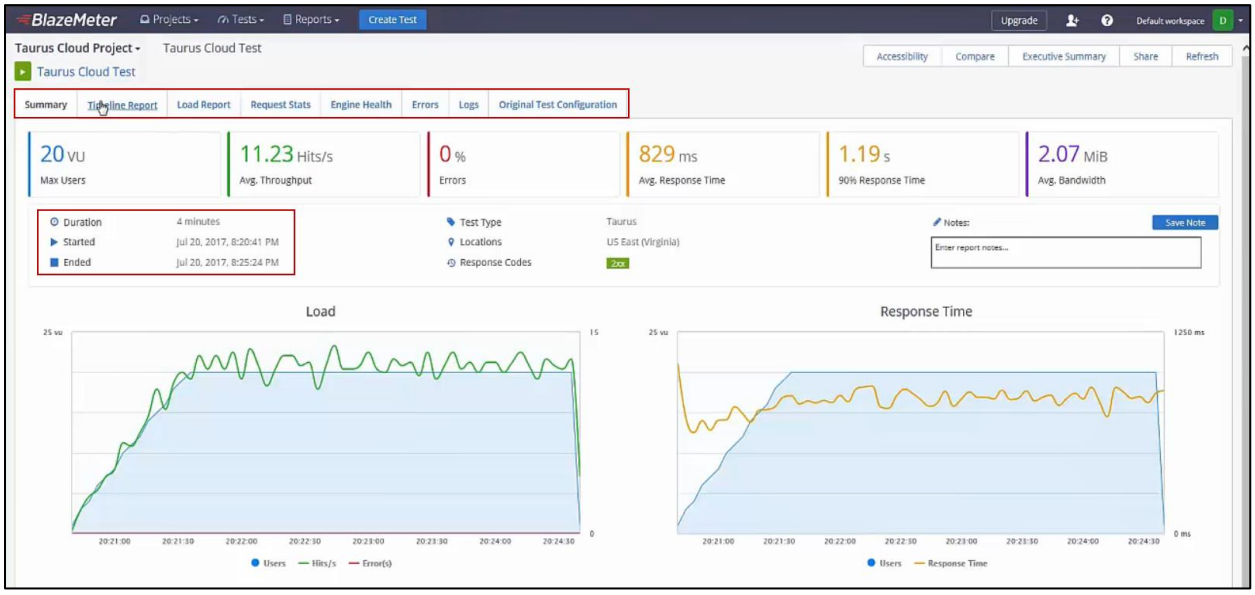


19. From the console, you can observe the number of actives users, cumulative statistics, number of hits on each label (Blazedemo.com and Blazedemo/reserve.php) that you have specified in your JMeter YAML script, and the failure percentage. As the load is generated from the cloud location, you can observe that details in console get updated slowly.

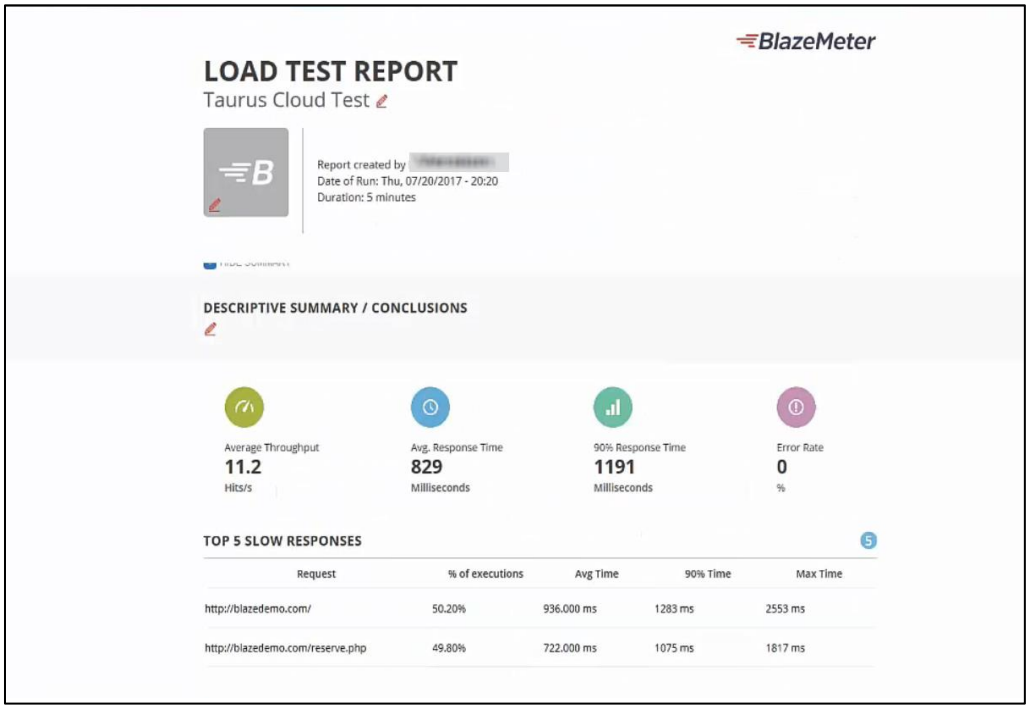


20. From the report, you can observe the timestamp as when the test has started and ended. You can navigate through different tabs in BlazeMeter UI to view the results. Unlike Taurus, the report on BlazeMeter is permanent, and you can access it whenever required.

Report Type	**Description**
Summary Report	Gives the dashboard view of your test while it is running and after it has finished. It appears as soon as your test starts to collect data.
Timeline Report	Enables you to view many different types of KPIs within one graph.
Load Report	Displays values for all request made during the test and an individual row for each named request in your test.
Engine Health Report	Displays the performance indicators received during the test while monitoring JMeter console(s) and engine(s).
Errors report	Contains the errors that were received by the web-server under the test because of HTTP request.
Original Test Configuration Report	Displays the complete load configuration properties along with session ID This ID will be used as a parameter in the API to identify a specific session of your BlazeMeter test.



21. Click the **Executive Summary** report tab in BlazeMeter UI. You can write your summary and conclusions in this section at the top of the report. This report displays Top 5 slow responses, Top 5 errors, Test setup details, summarized aggregate report, Summarized Error report, Graphs presenting users, Response times, and hits per second.



22. This concludes the demonstration.

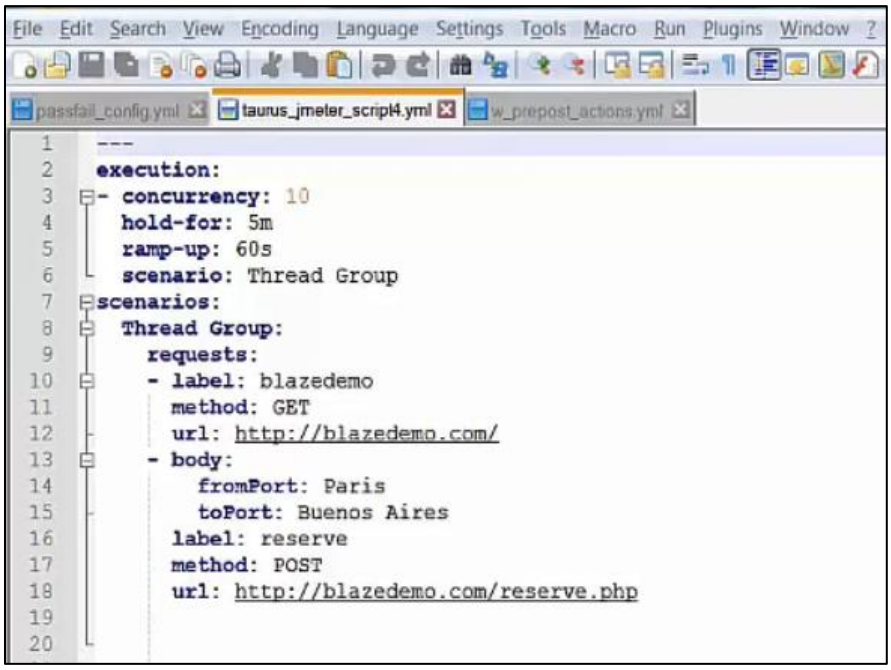
Lab – Execute Taurus Script with Pass/Fail Configurations and Pre/Post Actions

Goals	Execute a Taurus script with Pass/Fail configurations and Pre/Post Actions.
Scenario	Test Failure criteria for tests metrics and KPIs are an inherent part of performance tests since they ensure your product is performing as per the requirements. You can monitor the failure criteria manually to check if the metrics you want to measure have breached your threshold. However, this may be a time-consuming and effort intensive exercise. You can use the Taurus pass/fail module, to automate this process if one of your tests, meets the threshold criteria set by you, it will fail automatically, and you will get a notification in the report and logs. Taurus uses the "Shell Executor Service Module" that enables you to divide your test script into different logical steps and set your desired pre/post test actions in a single configuration file. In this demonstration, you will: ▪ Write a simple YAML script ▪ Set the pass/fail criteria in a script ▪ Set the pre/post test actions ▪ Execute the script ▪ View results in Taurus console
Time	10 minutes

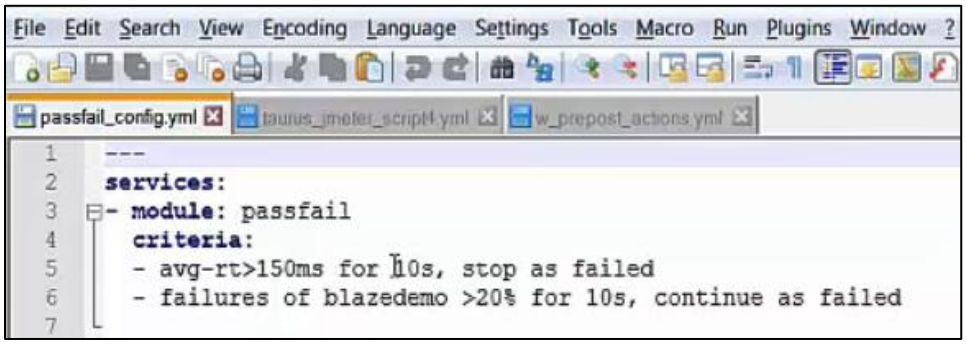
Instructions:

Download the lab resources and save it on your local system. You can directly use these JMX, YAML and Python scripts to run the labs in future.

1. Open the **taurus_jmeter_script4.yml** from the lab resources folder on your local system. This is the Taurus JMeter script written in YAML.



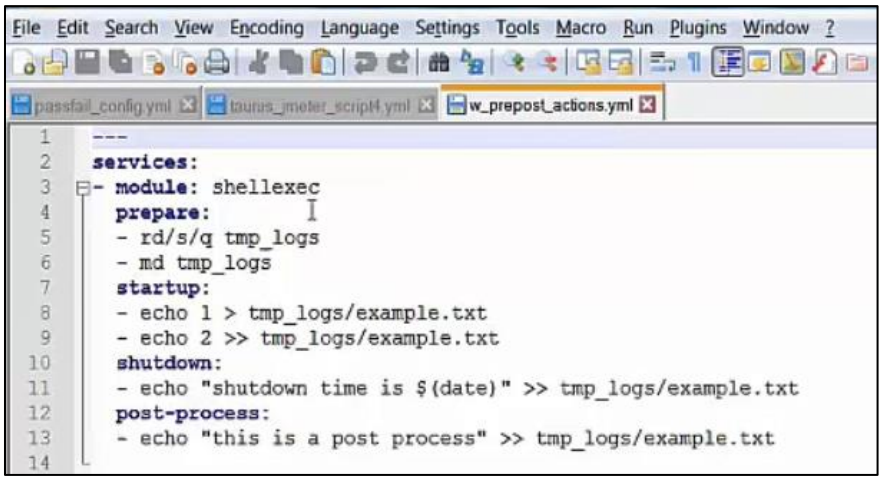
2. Open the **passfail_config.yml** file from the lab resources folder. In this file, you will define the pass/fail criteria for the test. Here, if the average response time is greater than **150ms for 10 seconds**, the test must **stop as failed.** If the hits are higher than **20% for 10 seconds** for the reserve label, the test must **continue as failed**.



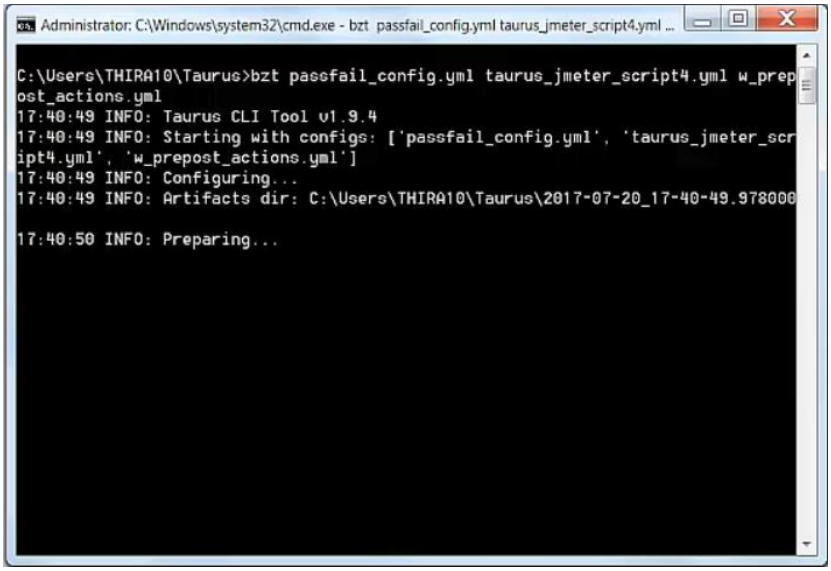
3. Open the file **w_prepost_actions.yml** from your lab resources folder. This file defines the pre and post actions implemented during Taurus script execution. You can execute shell commands such as

   - **rd/s/q tmp_logs**: It deletes the temporary logs directory
   - **Md tmp_logs**: It creates a temp log directory and writes the values into that files that you define.

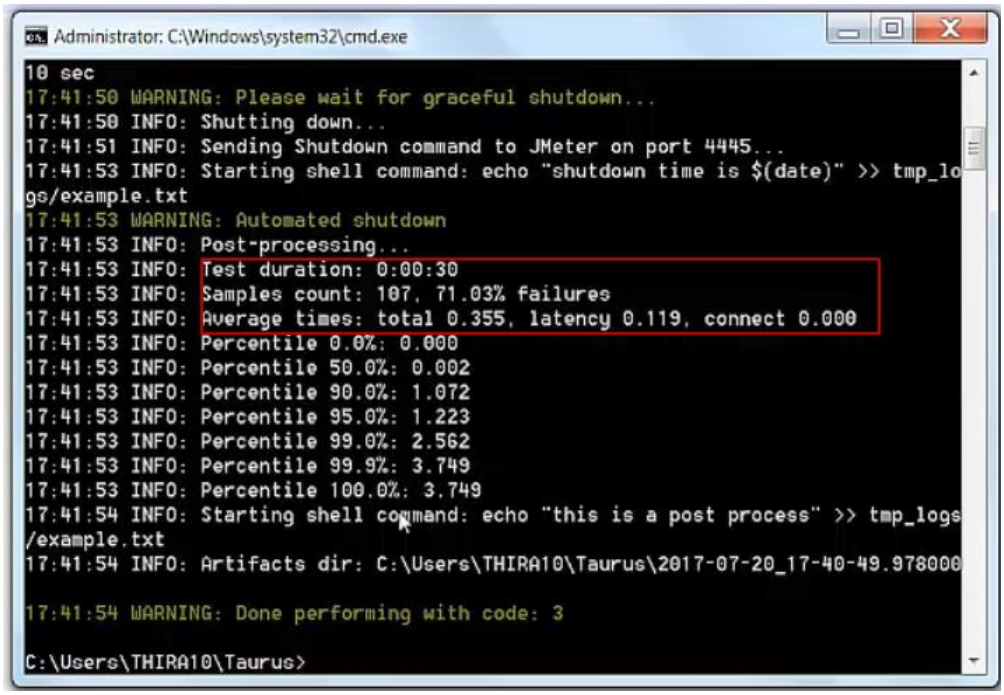
   Note: On execution, the tmp_logs directory is created in your system, and the values **1** and **2** are written into the example.txt file.



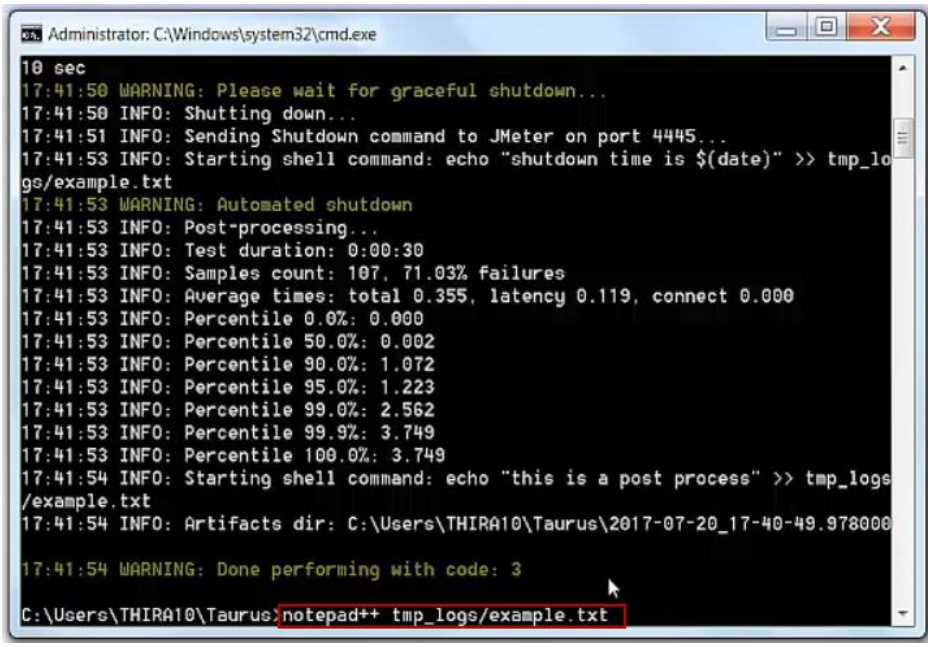
4. Run the bzt command with all 3 YAML scripts. Enter **bzt passfail_config.yml taurus_jmeter_script4.yml w_prepost_actions.yml** in command-prompt, and press Enter. The shell command execution starts as a pre-action for the test.



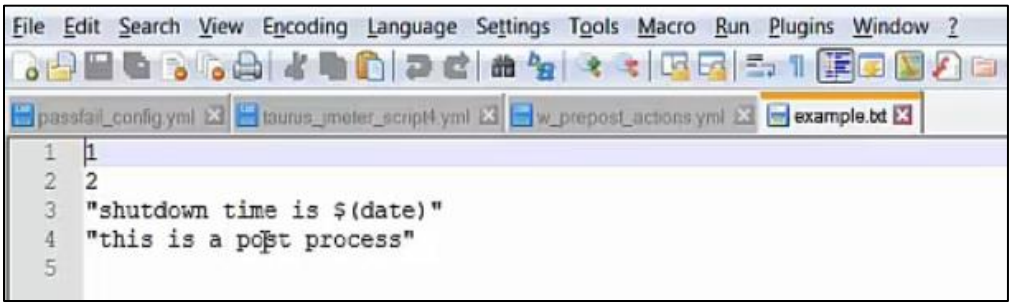
5. The Taurus console opens. Now, observe the **Failures** on the console for the **blazdemo** label. As per the script, if the failure for blazdemo.com is greater than **20% for 10 seconds**, the console automatically shuts down and returns to the command line.



6. Now open the example.txt file to review the pre/post actions. It is created under the **tmp_logs** directory. Type **notepad++ tmp_logs/example.txt** in command-line tool and press Enter.



7. The command executes successfully, and values are returned as defined in the **w_prepost_actions.yml** script.



8. You may change the pass/fail thresholds and simulate some failures. Review the log files and console reports for better analysis of your test execution. This concludes the demonstration.