opentext™

GUIDE

Hands-on Chaos Testing with OpenText Professional Performance Engineering

Take advantage of OpenText Professional Performance Engineering for all of your Chaos Engineering needs. Read now for a step by step guide on how to get started with the new integration.

Contents

A distributed services approach entails countless variables that even the most meticulous development team cannot predict. Even when each service functions perfectly in a vacuum, interactions between them can be—in a word—chaotic. So, to most effectively build infrastructure resilience against the unpredictable, test engineers developed chaos engineering.

This disciplined form of software engineering prioritizes the knowledge that failures are inevitable and that systems can be designed to withstand them without significantly affecting the end-user experience—even at scale. A primary component of this process is chaos testing, which simulates failures in a system so that developers can build resilience for the broadest possible array of failure scenarios.

To further explore this concept, this article will walk through using Gremlin to add some chaos to an application and OpenText™ Professional Performance Engineering to analyze the application's ability to handle the failures.

Getting Started with OpenText Professional Performance Engineering

To follow this tutorial, ensure that you have the following prerequisites. Additionally, note that these services are currently compatible with Windows only.

- OpenText Professional Performance Engineering (free trial available)
- Gremlin
- The OpenText™ VuGen tool for simulating typical user behavior on an application
- A web application to test. This tutorial uses this Spring Boot application, but you can use any app to which you have the rights.

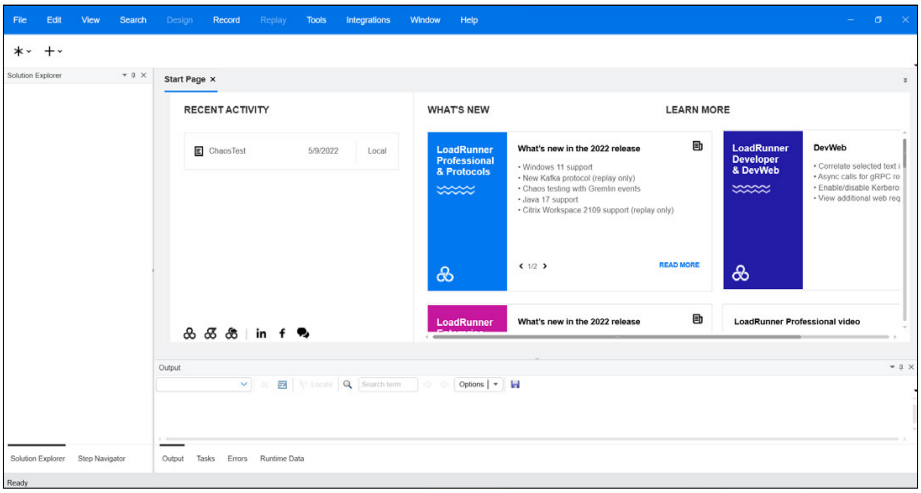
Before proceeding, follow these steps to install and configure Gremlin on your machine.

Adding Chaos Testing

Setting Environment for Chaos Testing

To get started, you'll need to enable OpenText Professional Performance Engineering to simulate virtual users using the OpenText Professional Performance Engineering VuGen (Virtual User Generator) script.

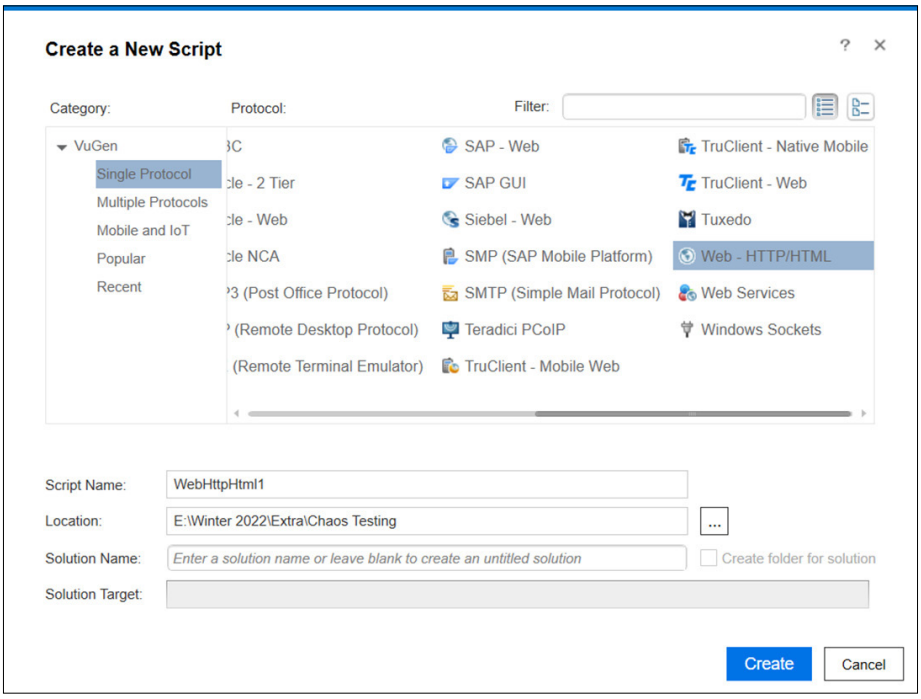
Open the VuGen App on your Windows system. Click the + symbol at the upper-left corner and select the Add New Script option.



From the **Create a New Script** window, select the **Single Protocol** option under the **Category**: tab at the far left.

Then, under **Protocol**:, select the **Web - HTTP/HTML** option.

Next, provide the name and location of your script. Your screen should look similar to this:

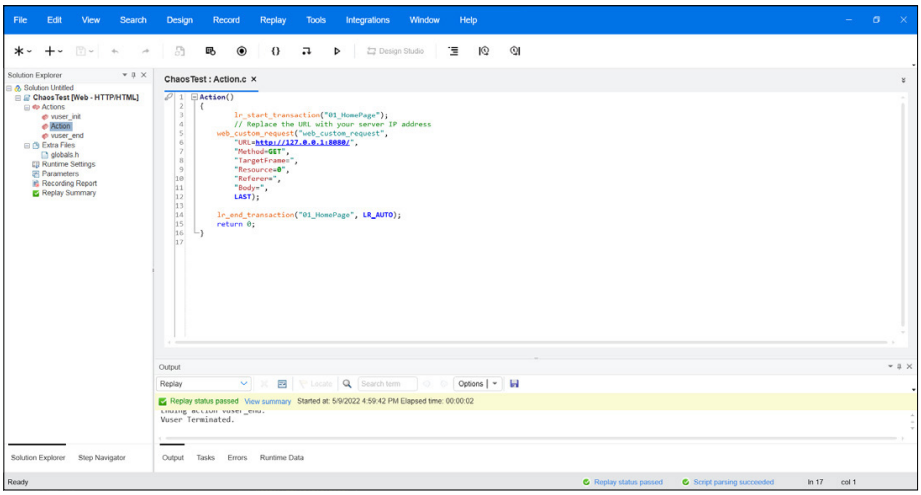


Once you've filled in the required fields, click the **Create** button.

If the application doesn't do so automatically, use the **Solutions Explorer** pane to navigate to **Chaos Test > Actions**. Then, double-click on **Action** to add the code below. Note that the specified URL is for the localhost, as this is where the demo web application is hosted.

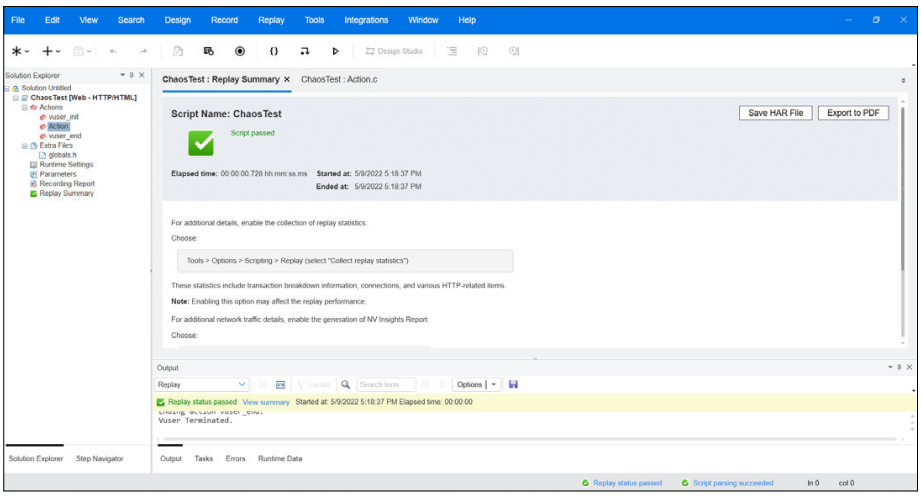
```
Action()
{
lr_start_transaction("01_HomePage");
// Replace the URL with your server IP address
web_custom_request("web_custom_request",
"URL=http://127.0.0.1:8080/",
"Method=GET",
"TargetFrame=",
"Resource=0",
"Referer=",
"Body=",
LAST);

lr_end_transaction("01_HomePage", LR_AUTO);
return 0;
}
```

Your VuGen window should now look similar to this:



The OpenText Professional Performance Engineering script above simulates a user-sent GET request to the specified URL.

Click the ▷ (Run) button on the toolbar to check whether the script runs without errors. If it runs successfully, VuGen will open a Summary tab similar to that in the following image. Otherwise, it will prompt you to resolve any discovered errors.



You've now successfully created a script that enables OpenText Professional Performance Engineering to simulate virtual users.

Creating a Gremlin Scenario for Chaos Testing

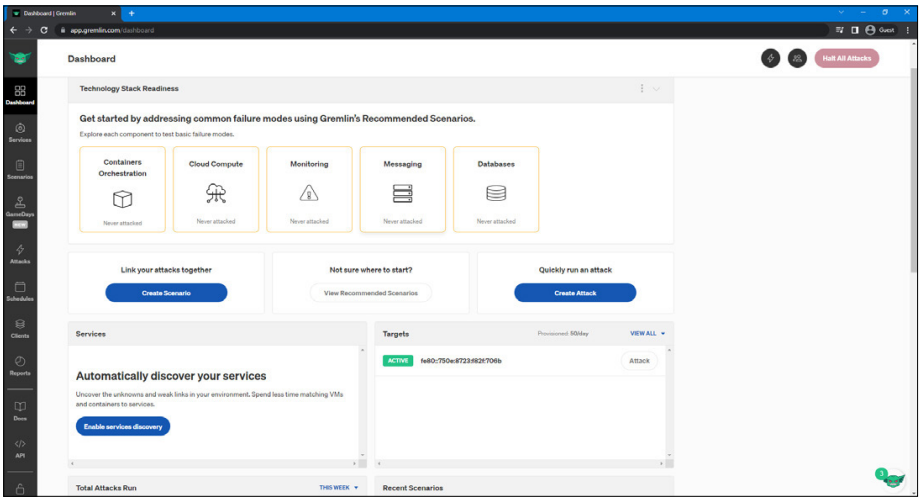
Next, for this demonstration, you'll create two attacks:

- A CPU attack to assess how the hosted web application performs when it uses nearly all available CPU resources.
- A latency attack to test the application's responsiveness over varying network conditions.

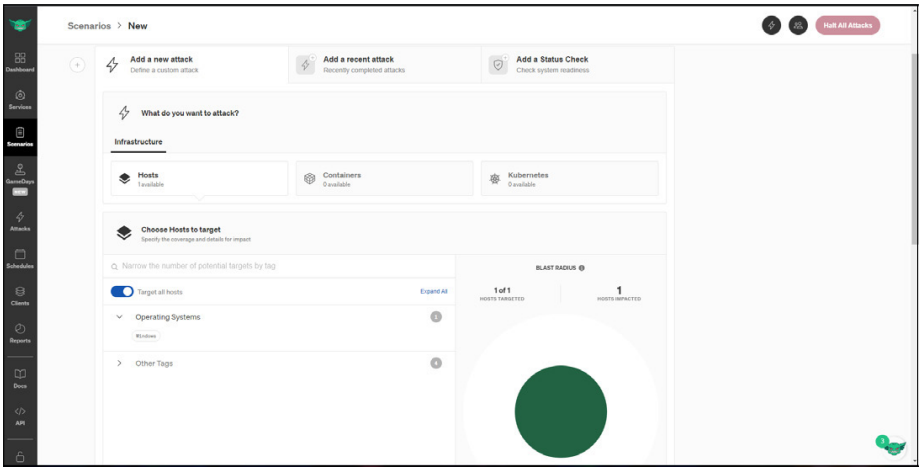
Once you've finished signing up for Gremlin, you need to create a team. This will help you access the Gremlin scenario using its Team ID and API key.

Note that you can view the installation documentation to ensure that you've properly configured the application for Windows.

Now, click on **Dashboard** from the left-hand sidebar. Then, click the **Create Scenario** button to begin creating a chaos testing scenario.



If you've configured your Gremlin app and teams correctly, you should be able to see the number of hosts connected to the server. The image below shows that one host is available to test the scenario on the Windows system:

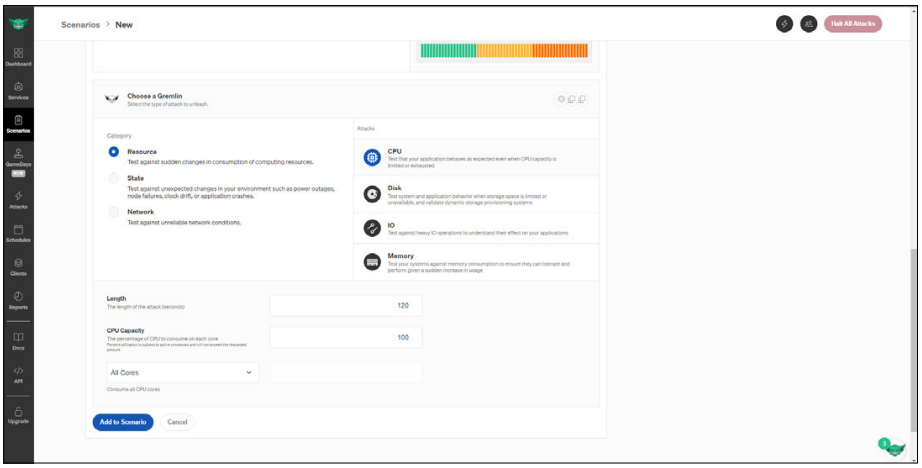


Click Add a new attack to open the Choose a Gremlin section.

Then, select the Resource radio button from the Category list and click CPU from the Attacks list.

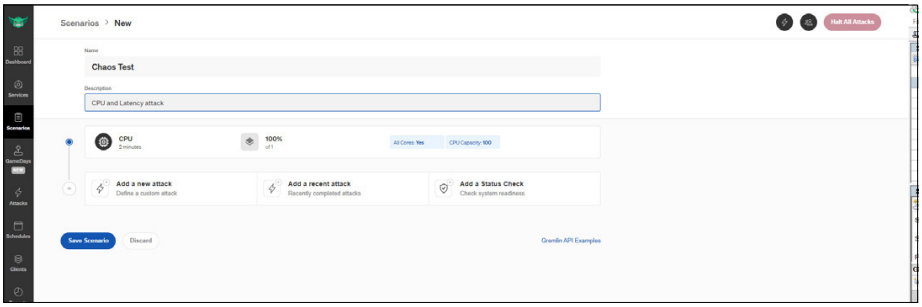
Next, set the Length of the attack to 120 seconds and ensure that the CPU Capacity value is set to 100.

Now, select All Cores from the dropdown menu to ensure all available cores are used for testing.



Finally, click the **Add to Scenario** button.

Your **Dashboard** should now appear similar to what's shown below:

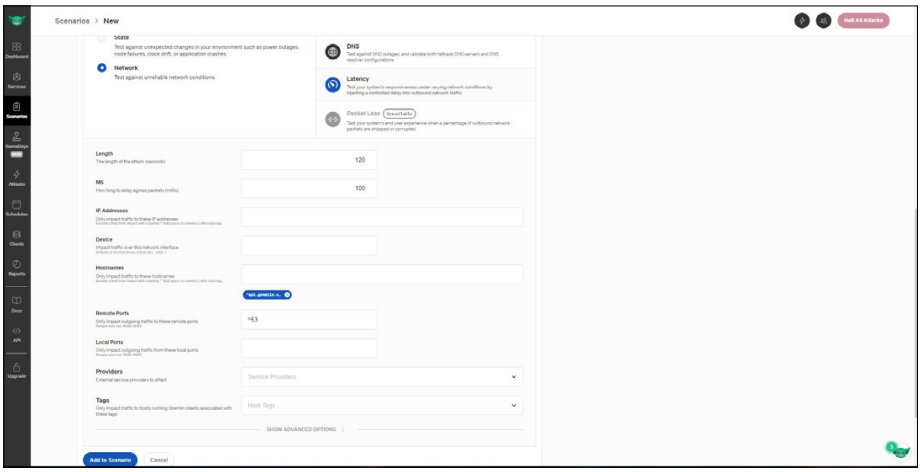


You can name your scenario and add a brief description, but don't save the scenario just yet—you'll now add the latency attack.

Once again, click **Add a new attack**.

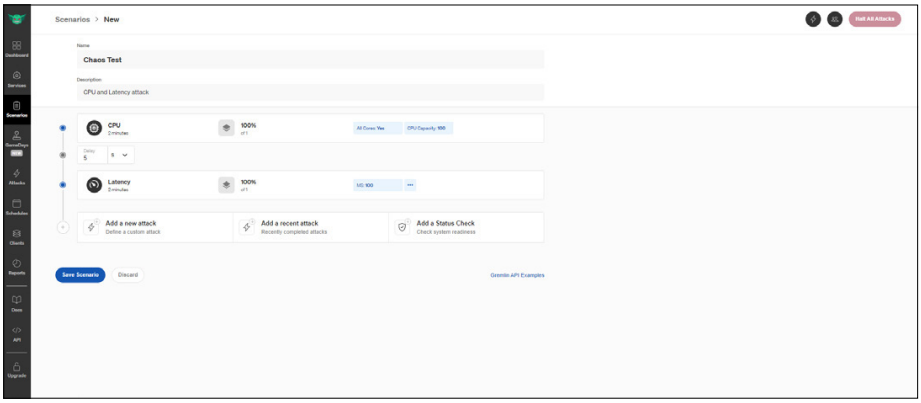
This time, select **Network** from the **Category** list and click **Latency** in the **Attacks** section.

Then, update the Length of the attack to 120 seconds and keep the delay at 100 **MS** (milliseconds). You can leave the rest of the fields blank or add optional details for your specific scenario.



Once finished, click the **Add to Scenario** button.

Your window should now look like the image below:



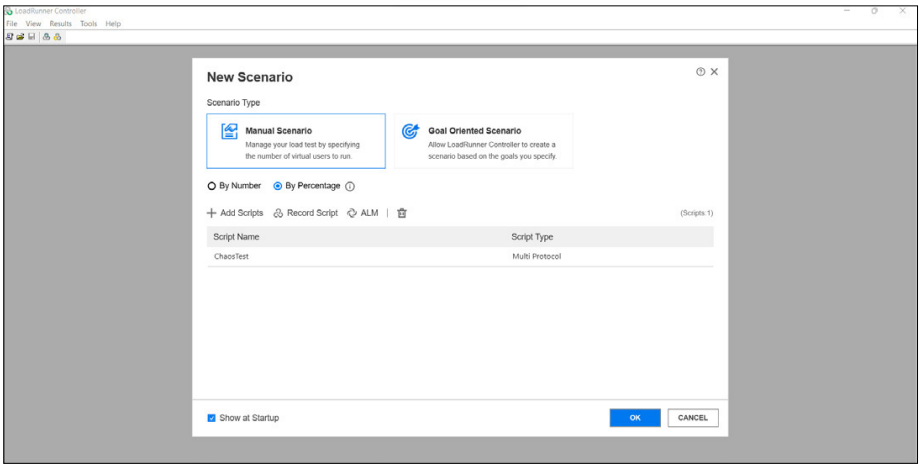
Finally, click the **Save Scenario** button. Now, when you click on the **Scenarios** button from the sidebar, you should see a section named for your created test scenario.

Performing the Chaos Test Using Controller

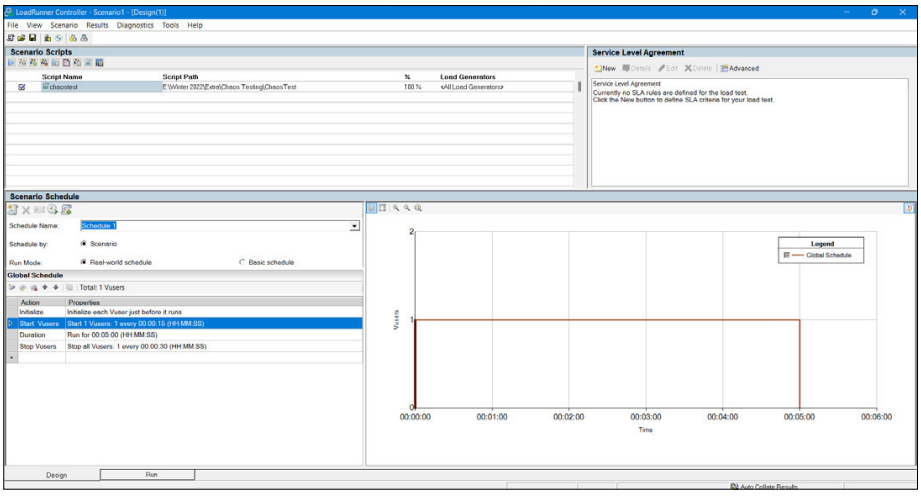
Now you're ready to execute your chaos test using Controller.

When you open Controller on your Windows machine, the program will open a **New Scenario** window. Select **Manual Scenario** under **Scenario Type**.

Next, add the VuGen script created earlier. Click **+ Add Scripts** and navigate to the folder where the script is saved. Select the script file and click the **ADD** button.



Once you've added the script, Controller will open it, as shown in the image below:



You'll now need to define a few criteria in the **Global Schedule** panel at the lower-left corner of the window.

First, double-click on the **Start Vusers** section and set the **Start** value to 50 Vusers. This is the maximum value that OpenText Professional Performance Engineering Community Edition permits.

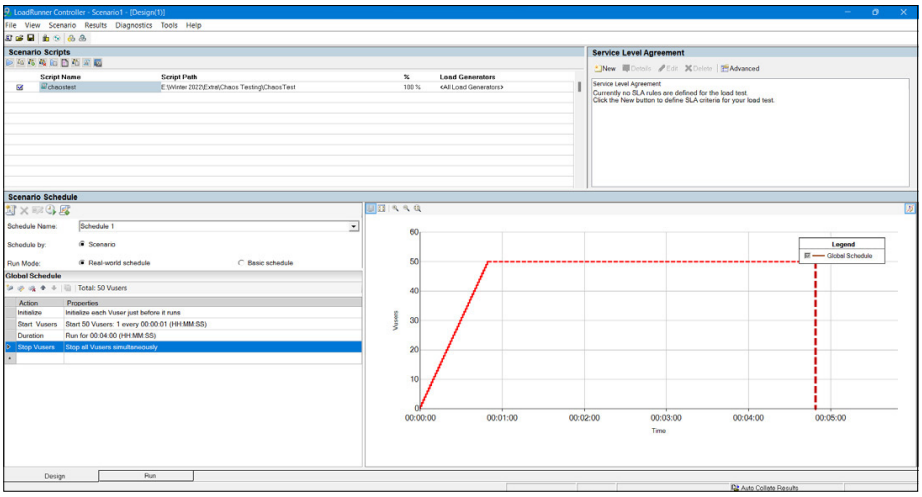
Additionally, set the **(HH: MM: SS)** parameter to 00:00:01. This represents one user logging in per second to maximize the chaos introduced into the system.

Then, click **Next** to move to the **Duration** field. Set it to 00:04:00. This means that the test will run for four minutes.

Next, navigate to the **StopVusers** field and select the **Simultaneously** radio button. This will create an event in which all users exit the app at the same time.

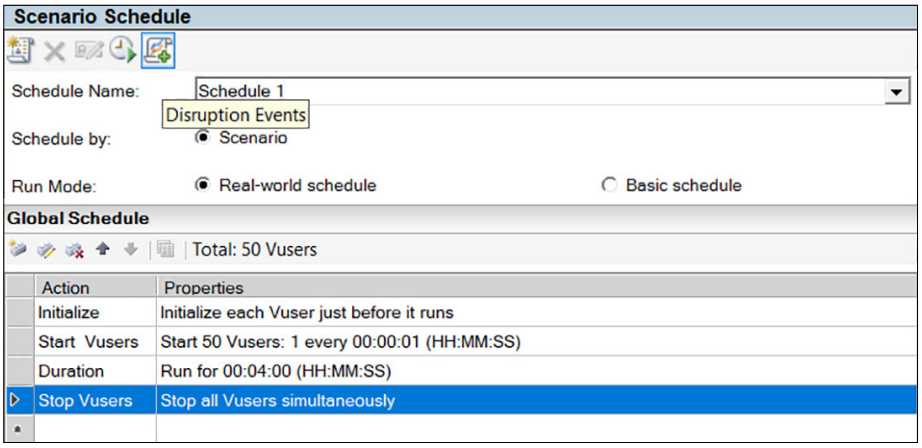
Finally, click the **OK** button.

You should now notice that the graph indicating Vusers versus Time has changed:



Note that since you're testing for chaos, you can configure the test to ignore think time. To do so, locate the checkbox for your script under the **Scenario Scripts** section in the upper-left corner of the window. Right-click the white space near the checkbox and navigate to **Runtime Settings**. Then, in the **General** section of the new pane, click on **Think Time**. Select the radio button for **Ignore think time**.

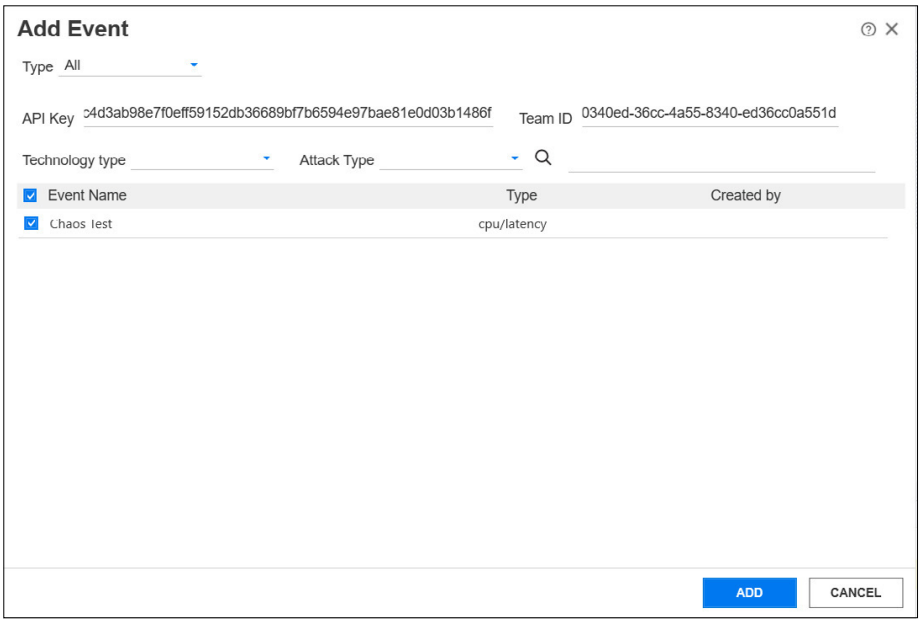
To add the chaos scenario created in the Gremlin app, click the rightmost button directly beneath the **Scenario Schedule** header. This is the **Disruption Events** button.



Then, click the **+AddEvent** button and provide the API key and Team ID of the team you created in your Gremlin account.

Once you add the Team ID and API key, you'll see the created scenario as shown in the image below.

Select the event and click **ADD** button to add it to the controller scenario. Set the **Start Time** to 00:01:00. This means that the disruption event will start one minute into the scenario.

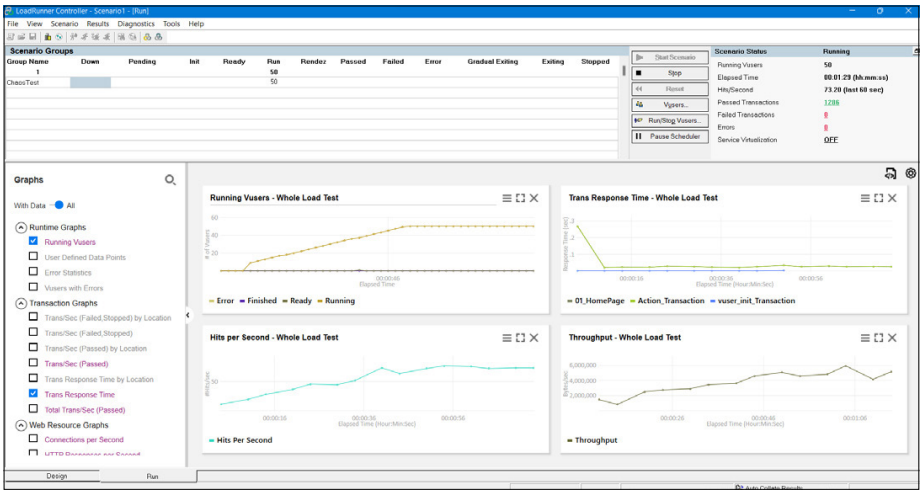


Next, click the floppy disk/save icon below the top menubar and provide a name for your saved scenario. The demonstration has kept the name as Scenario1.lrs.

Select your destination folder for the .lrs file and click **Save**.

Now, you can run the scenario by clicking the **Start Scenario** button toward the upper-righthand corner of the window.

OpenText Professional Performance Engineering will then produce graphs of the results and display them as shown below:

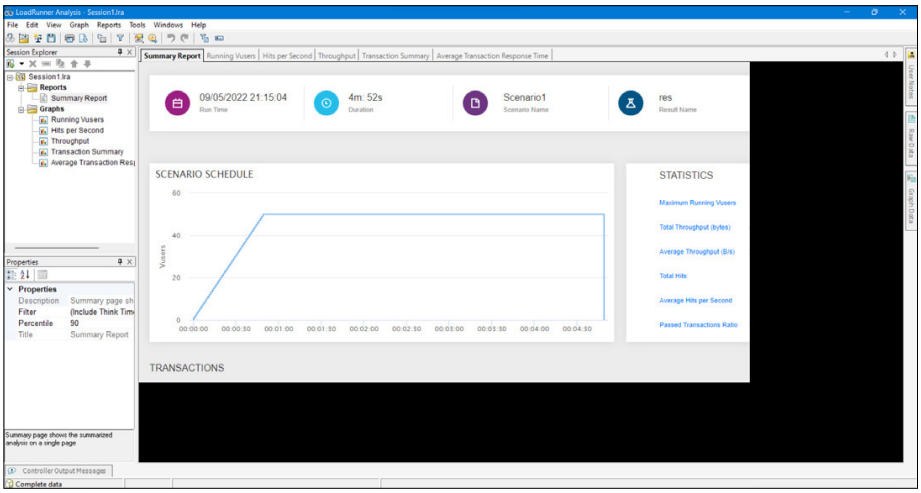


Test Results

Once the scenario has finished running, click the **Reports** option from the menubar and select **Analyze Results**.

Once all the summarized results have been generated, you'll receive a pop-up to view them.

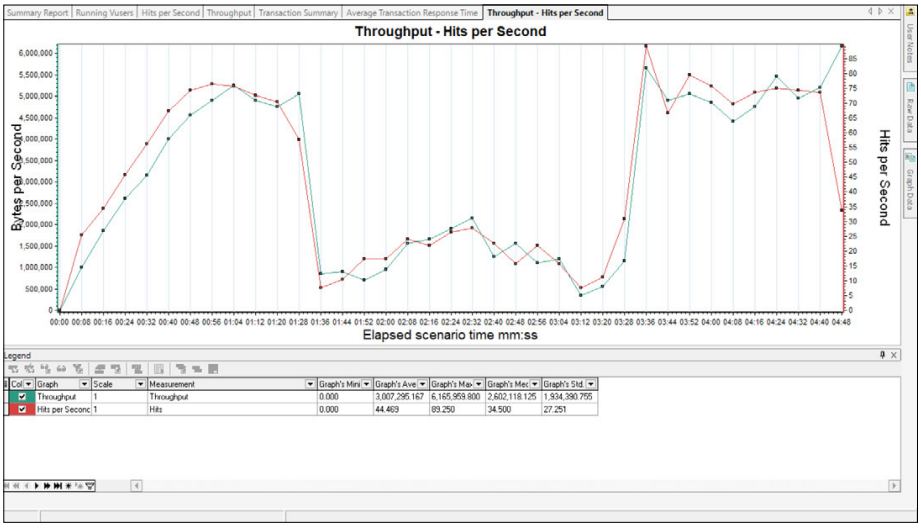
Click **Yes**. This will open graphs displaying **Hits per Second, Throughput, Average Transaction Response Time**, and several other statistics.



You can now compare the output data.

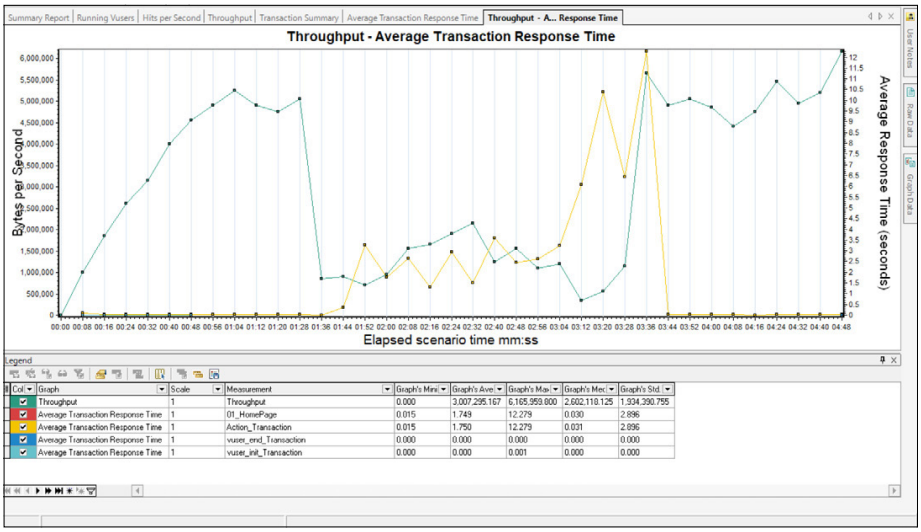
Click on the **Throughput** graph, then right-click on the chart.

Then, select the **Merge Graphs** option and choose the **Hits per Second** graph from the dropdown. Click **OK** to see the detailed diagram as shown in the image below.



The chart indicates that for the first minute, the application was up and running with high throughput and hits. Then, after the chaos started, it dipped sharply around 01:28, seesawing dramatically over the next 90 seconds, at which point the chaos event ends.

Merging **Throughput** and **Average Response Time** shows an inverse relationship between throughput and response time because of the chaos:



Your results will likely vary depending on your system configurations and scenario values. Still, the detailed analysis of how well the configured app handles the chaos is invaluable in a production environment. It can enable you to determine what triggers which failures and develop solutions to manage similar real-world events.

Conclusion

Chaos engineering is crucial for preventing the breakdowns of services that can eventually lead to full application failures. Now that you've experienced some of the capabilities that chaos testing can provide, you can more clearly identify ways to integrate it into your development and testing cycles. As organizations continue to shift to distributed architectures, the benefits of this engineering, testing, and analysis make the practice a principal component in building resilience within chaotic systems.

Learn more at

www.opentext.com/products/loadrunner-professional

www.opentext.com/products/devops