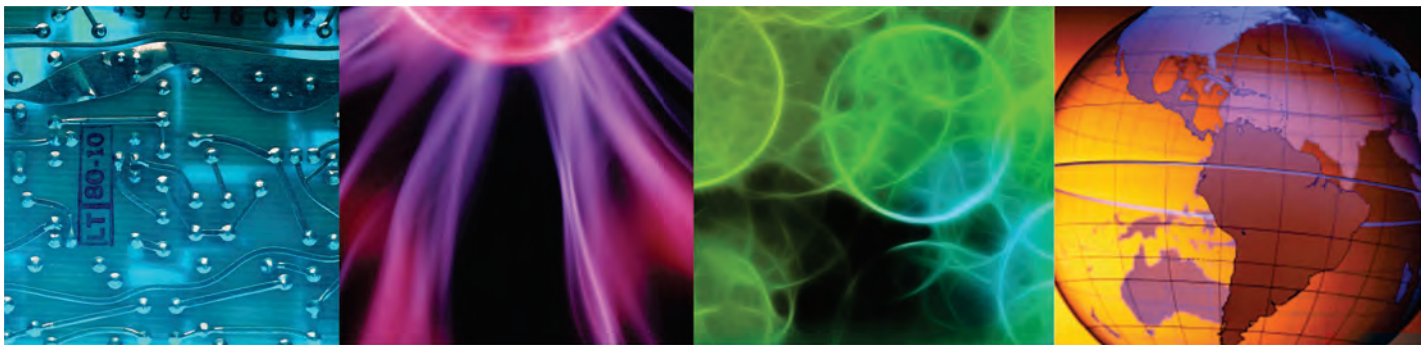
IBM Training

IBM Application Performance Management Advanced 8.1.3 Fundamentals

Course Exercises

Course code TOD45 ERC 1.0

August 2016



ibm.com/training

Authorized IBM Training

Contents

About these exercises

One user ID and password combination is used throughout this class: **root** user with password **object00**.

1 Monitoring with IBM Application Performance Management Advanced

IBM Application Performance Management Advanced includes monitoring agents for several application platforms. As of this writing, four of those monitoring agents support code-level monitoring:

- WebSphere® Application Server
- Microsoft .NET
- Node.js
- Ruby

Two of those agents also supply application topologies:

- WebSphere Application Server
- Microsoft .NET

Several other agents that are included in IBM® Application Performance Management Advanced support transaction tracking and synthetic transaction monitoring. Two of those agents are included in this course:

- Response Time Monitoring agent
- Synthetic Transactions agent

Objectives

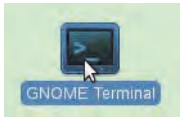
After completing this unit, you can perform the following tasks:

- Start DB2
- Modify the gdc_custom.properties file
- Start the DayTrader script
- Access the Performance Management console

Exercise 1 Starting DB2

You now start DB2® on **LIN1**.

1. Double-click the **GNOME Terminal** icon to open a command prompt.



2. Type this command and press Enter.

   `su -- db2inst1`

3. Type this command and press Enter.

   `db2start`

```
lin1:~ # su -- db2inst1
db2inst1@lin1:/root> db2start
```

4. After DB2 starts, type **exit** to return to the root user.

Exercise 2 Modifying the gdc_custom.properties file

You now modify the **gdc_custom.properties file** to increase the amount of data that is collected and displayed in the **Performance Management console**.

You change three parameters:

- To collect diagnostic data for every request, set the property to false.

  `dc.sampling.enable=false`

- To collect method data for every request for which diagnostic data is collected, set the property to false.

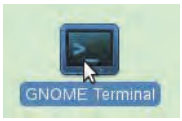
  `dc.sampling.methsampler.enabled=false`

- To enable method data collection at server startup, set the property to true.

  `dfe.enable.methoddata=true`

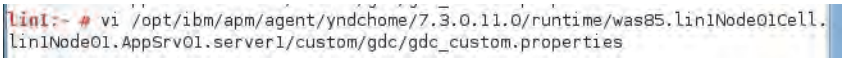
Note: The instructions are based on using **vi**. Feel free to use another Linux-based text editor and adapt the instructions.

1. On **LIN1**, double-click **GNOME Terminal** to open a command prompt.



2. Type this command (on one line) and press Enter.

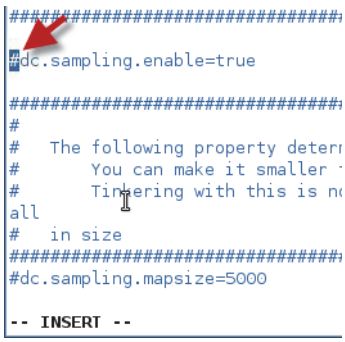
   `vi /opt/ibm/apm/agent/yndchome/7.3.0.11.0/runtime/was85.lin1Node01Cell.lin1Node01.AppSrv01.server1/custom/gdc/gdc_custom.properties`



3. Type **/dc.sampling.enable** and press Enter to find the parameter.

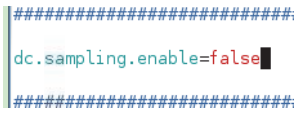
4. If necessary, use the arrow keys to move to the correct line in the file.

5. Type **Shift-I** to enter **Insert** mode at the head of the line.



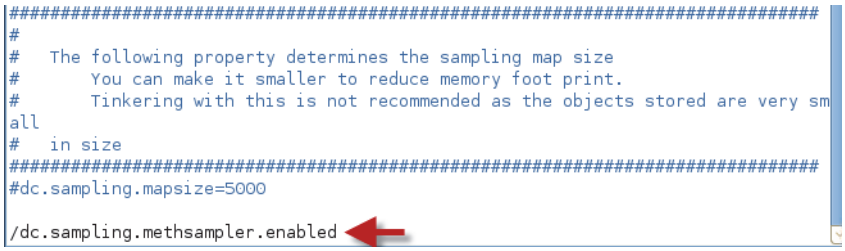
6. Delete the # character**.**

7. Press **End** to move to the end of the line and set the parameter to **false**.

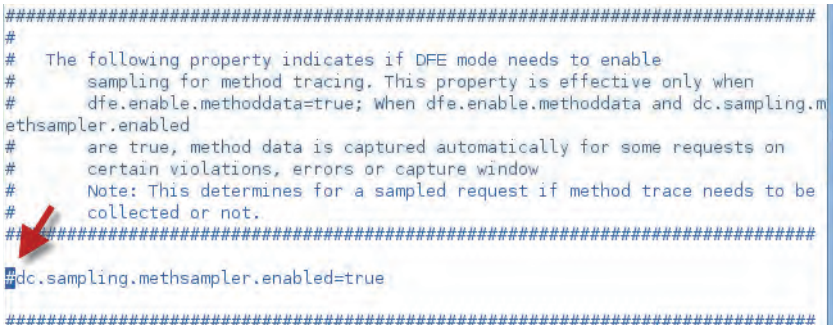


8. Press **Esc** to exit **Insert** mode.

9. Type **/dc.sampling.methsampler.enabled** and press **Enter** to find the parameter.



10. If necessary, use the arrow keys to move to the start of the line.



11. Type **Shift-I** to enter **Insert** mode at the head of the line.

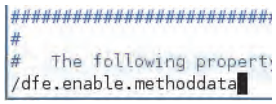
12. Delete the # character**.**

13. Press End to move to the end of the line and set the parameter to **false**.

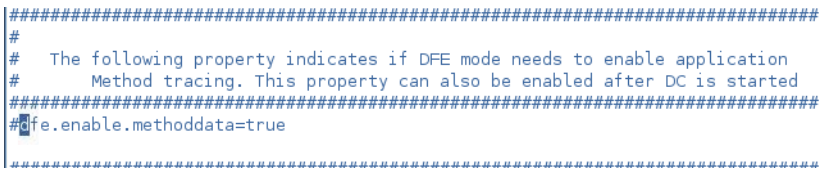
14. Confirm the parameter value.

15. Press Esc to exit **Insert** mode.

16. Type **/dfe.enable.methoddata** and press Enter to find the parameter.



17. If necessary, use the arrow keys to move to the start of the parameter.



18. Type Shift + I to enter **Insert** mode at the head of the line.

19. Delete the # character**.**

20. Change the parameter value from **false** to **true**.

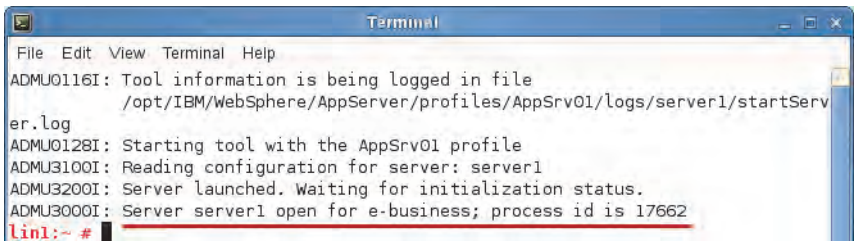
```
##############################
gfe.enable.methoddata=true

##############################
```

21. Press Esc to exit **Insert** mode.

22. Type `:wq!` to save your changes and exit the file.

23. Right-click the **Start WebSphere** desktop icon and select **Open**.

24. Wait until the **Server server1 is open for e-business** message in the terminal window.



25. Type the following command to stop the WebSphere agent. Wait for the operation to complete before proceeding.

    `/opt/ibm/apm/agent/bin/was-agent.sh stop`

26. Enter the following command to start the WebSphere agent:

    `/opt/ibm/apm/agent/bin/was-agent.sh start`

```
lin1:~ # /opt/ibm/apm/agent/bin/was-agent.sh stop
Processing. Please wait...
Stopping Monitoring Agent for WebSphere Applications ...
Monitoring Agent for WebSphere Applications was stopped with force.
lin1:~ # /opt/ibm/apm/agent/bin/was-agent.sh start
Processing. Please wait...
Starting the Monitoring Agent for WebSphere Applications...
Monitoring Agent for WebSphere Applications started
lin1:~ #
```

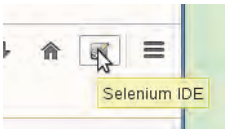
Exercise 3 Starting the DayTrader script

With IBM HTTP Server Response Time plug-in, top 10 resource timings are collected by default if your browser has JavaScript enabled. The Selenium IDE plug-in was installed in the Firefox browser on **lin1.ibm.edu** to capture traffic that the Response Time Agent can detect.

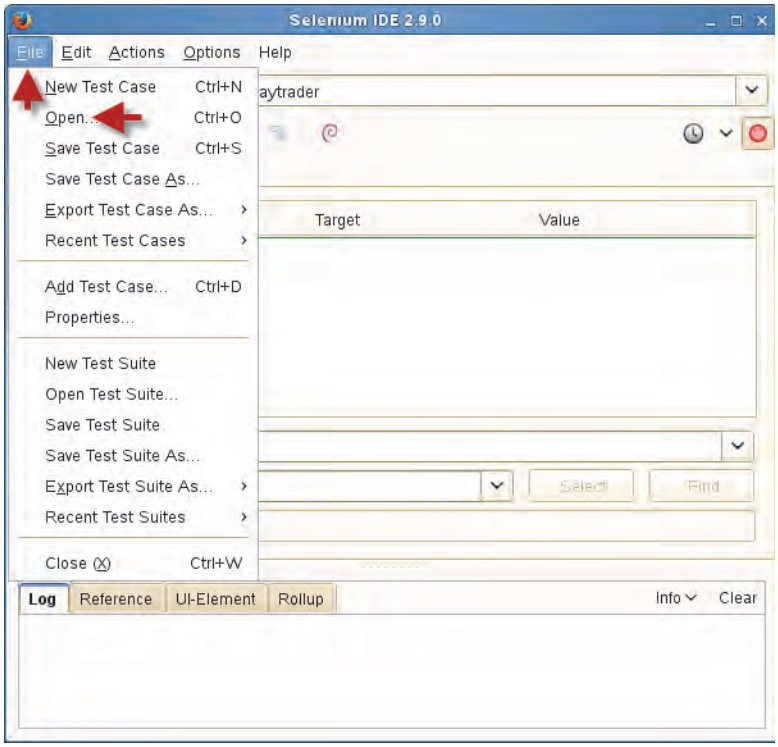
You now adjust the configuration of the application that is used for the WebSphere monitoring exercises.

1. Open a Firefox Browser on **LIN1** by double-clicking the Firefox icon on the desktop.

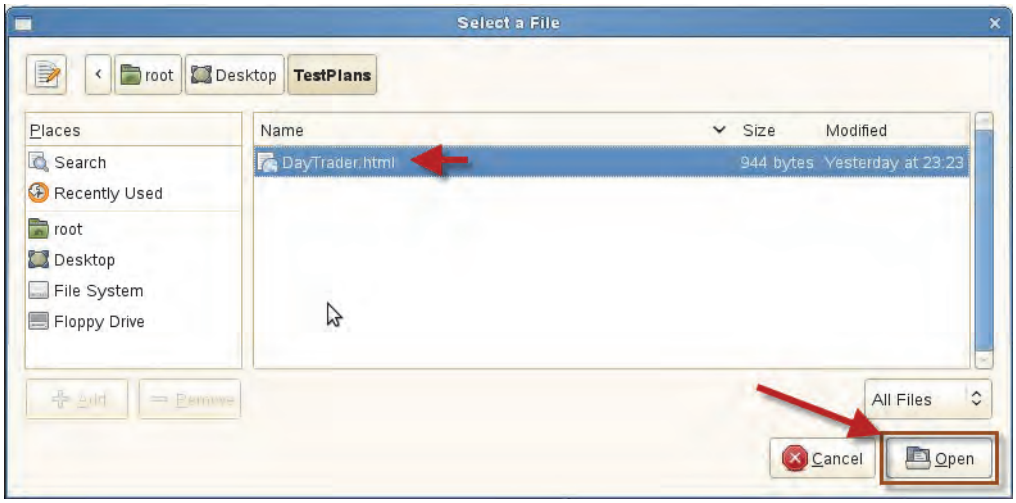
2. Click the Selenium IDE icon in the upper right of the browser window.



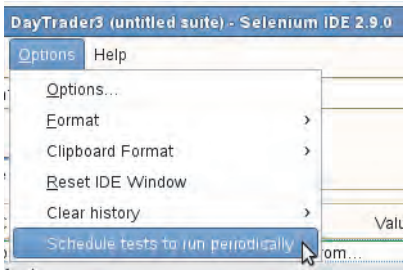
3. Click **File > Open**.



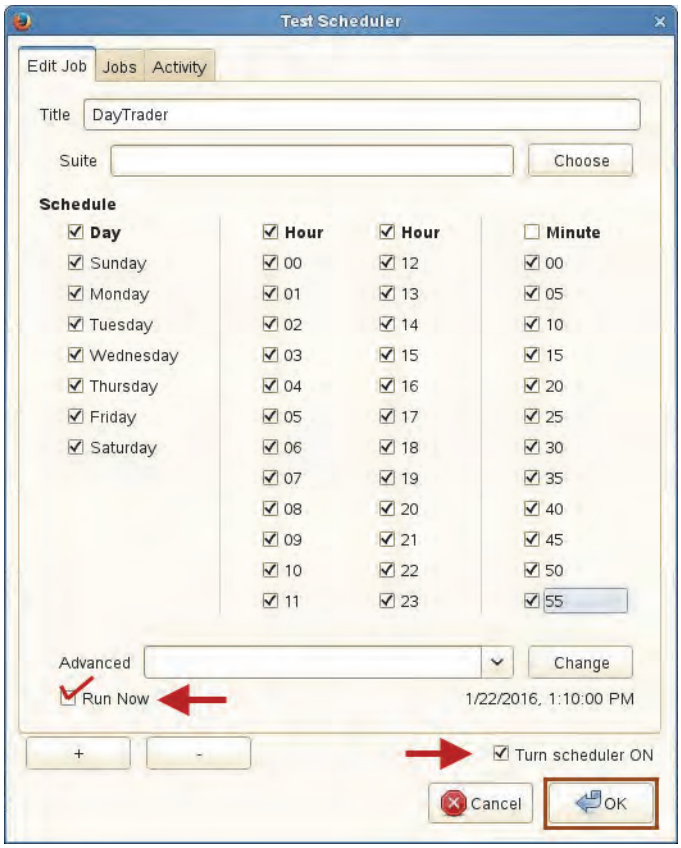
4. Select the **DayTrader** file in the **TestPlans** folder on the desktop and click **Open**.



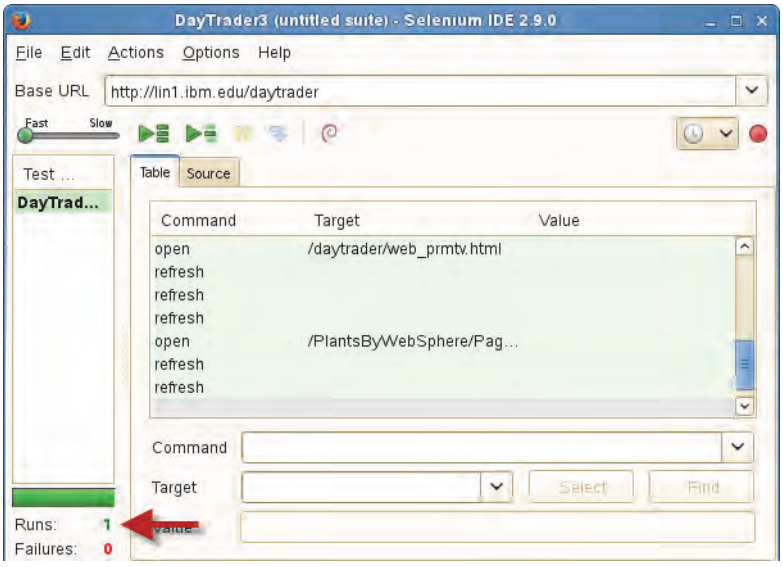
5. Select **Options > Schedule tests to run periodically** from the toolbar menu.



6. Select **Run Now** and **Turn scheduler ON**. Click **OK**.



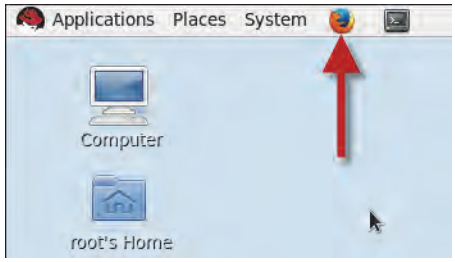
7. Confirm success of the script.



8. Proceed with the next exercise.

Exercise 4 Accessing the Performance Management console

1. On the **APM** VM, click the toolbar icon to open a Firefox browser.



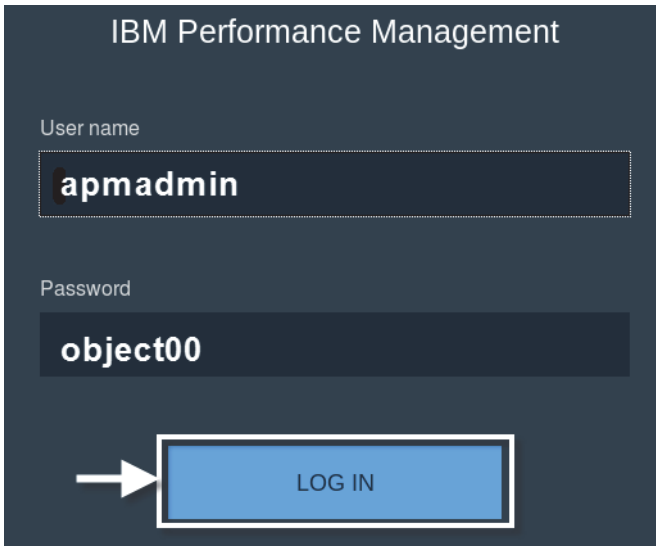
2. Open this web page:
   https://apm.ibm.edu:9443

3. Click **I Understand the Risks**.

4. Click **Add Exception**.

5. Click **Confirm Security Exception**.

6. Enter the user ID **apmadmin** and the password **object00**; click **LOG IN**.



7. Click **Remember Password**.

2 Application resource monitoring

In this lab session, you use IBM Application Performance Management to monitor WebSphere resources.

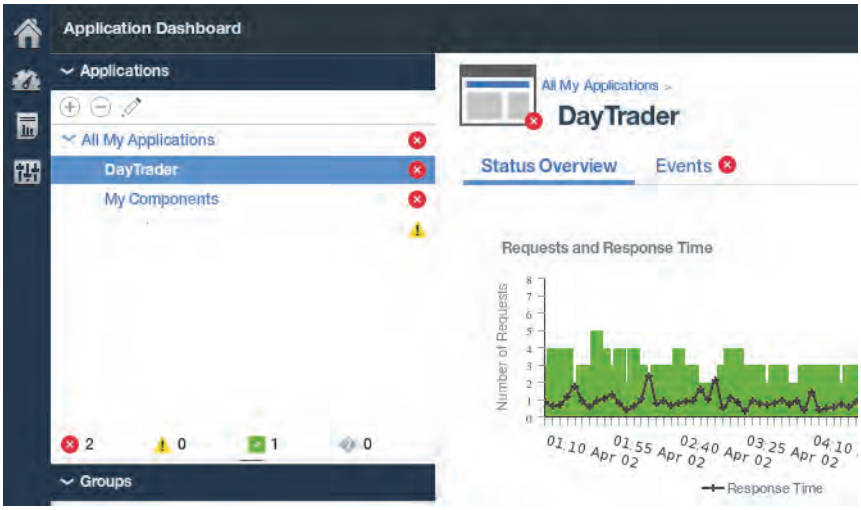
Objectives

After completing all the exercises, you can monitor WebSphere Application Server resources by using the Performance Management Console

Exercise 1 Monitoring WebSphere resources

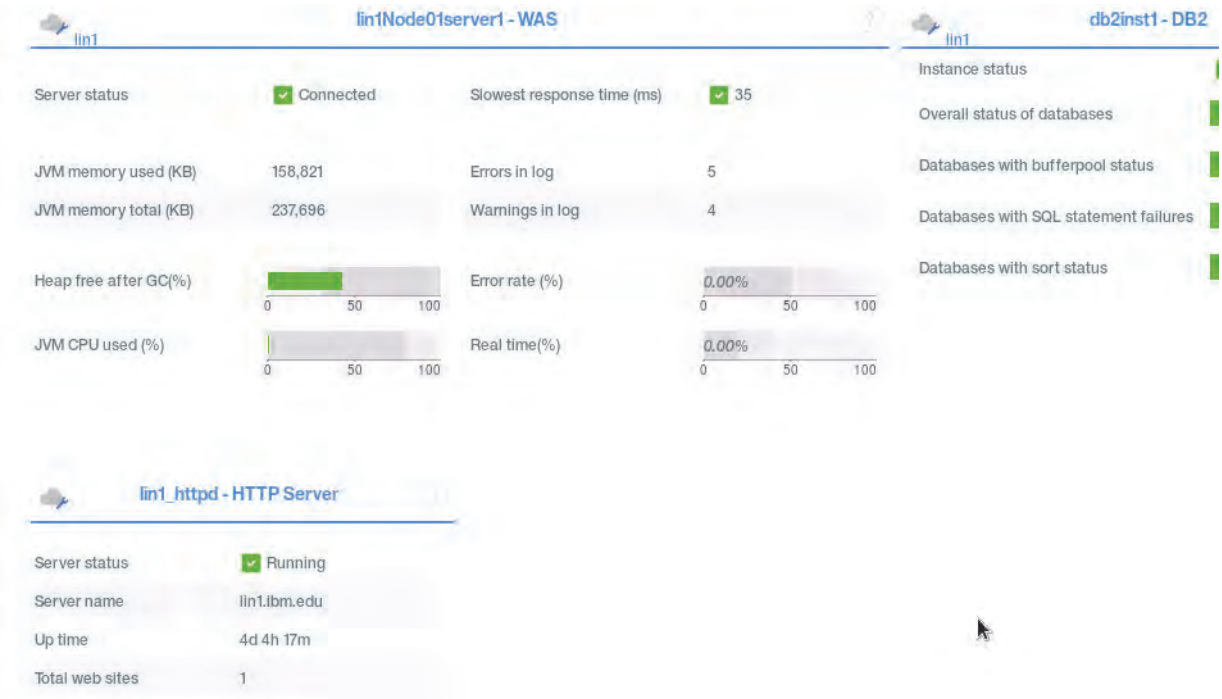
1. Hover over the speedometer icon, then click **Application Performance Dashboard** in the **Performance** menu.

2. In the **Performance Management** console, select **DayTrader** in the application list.

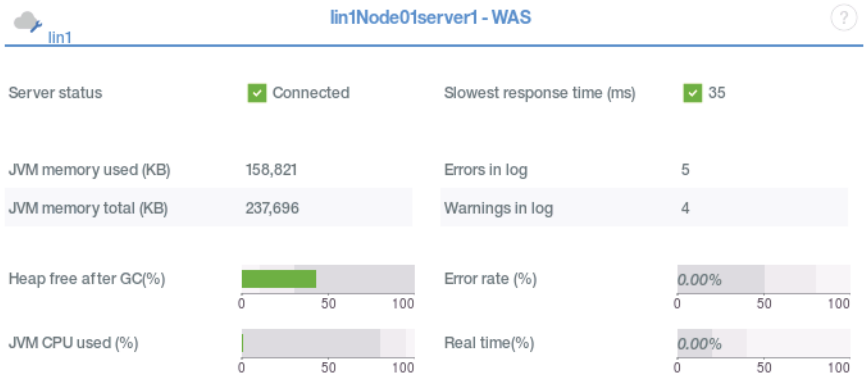


3. Click **Components** in the **Groups** widget.

4. Review the component widgets on the **Status Overview** dashboard.

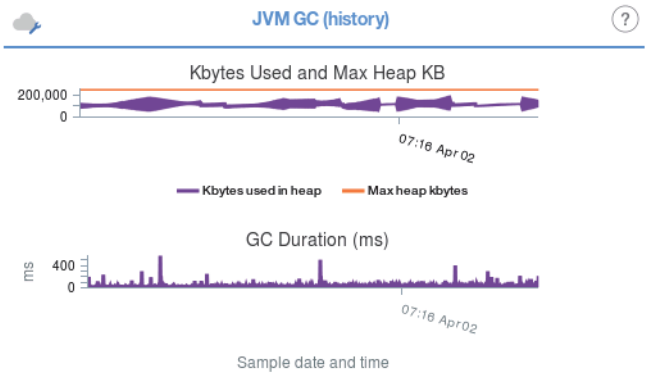


5. Click the **WebSphere** widget to access the resource dashboard.



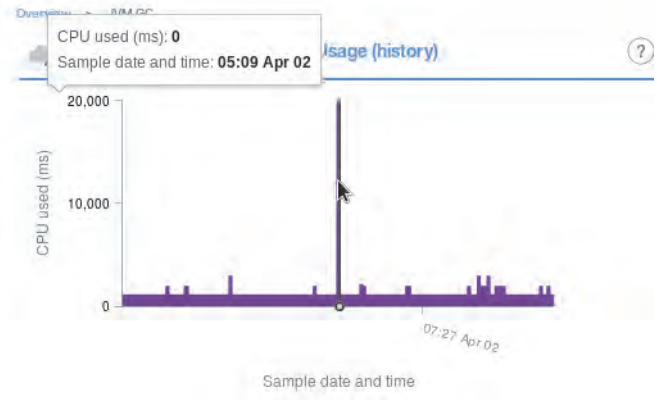
6. Review the resource widgets of the **WebSphere Status Overview** dashboard.

7. Click the **JVM GC (history)** widget to access the dashboard for garbage collection.



8. Review the dashboard widgets for **JVM** and **Garbage Collection**.

9. Hold down the left mouse button while hovering over one of the peaks in the **JVM CPU Usage (history)** widget to see the data point value and time.

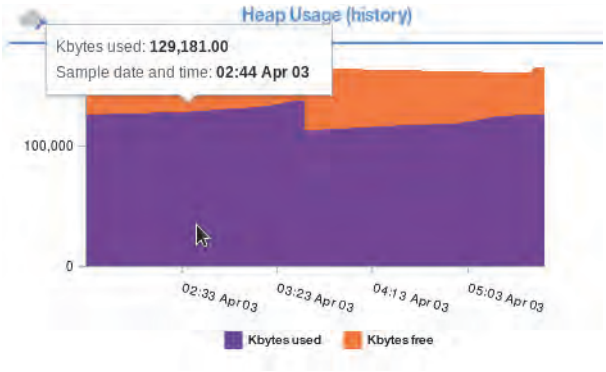


10. Perform the same action on the **JVM Garbage Collection** and **Throughput and Average Response Time** widgets.

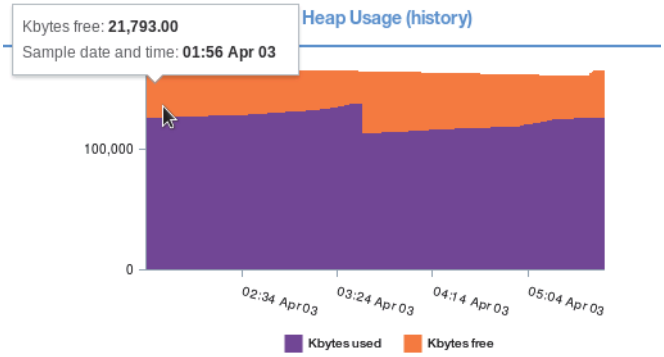
Note: On a lightly loaded lab system, none of the values in these charts are alarming. However, in a production environment, the ability to identify the precise time of a spike in CPU usage or Response Time is valuable.

For chart widgets with multiple data attributes, you can position the mouse over the display area of the statistic for which you want the value for a specific time.

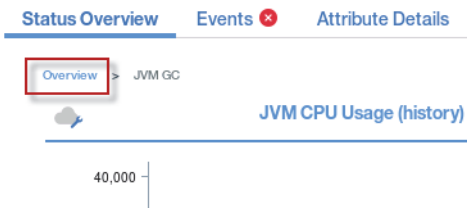
11. Move the mouse pointer over the purple data display area of the **Heap Usage (history)** chart to see values for the relevant attribute.



12. Move the mouse pointer over the orange data display area of the **Heap Usage** chart to see values for the relevant attribute.

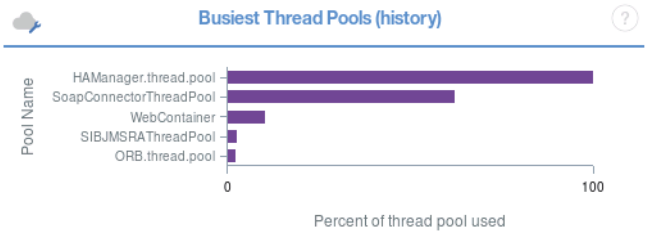


13. Just below the **Status Overview** tab, click the **Overview** link to return to the main WebSphere resource dashboard.



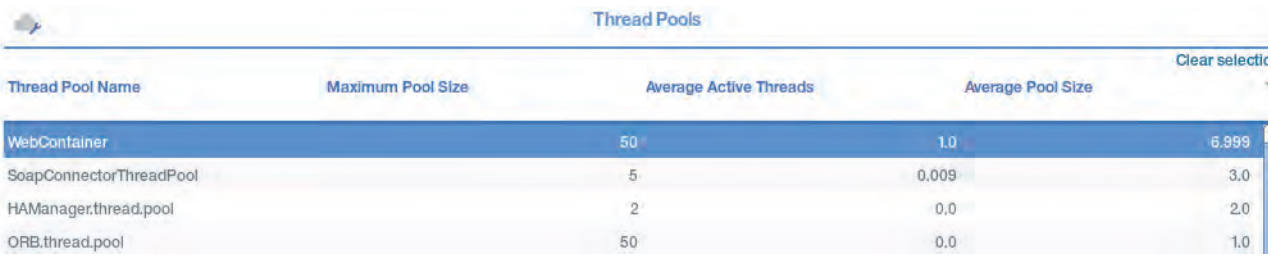
Note: No further dashboards are available for JVM garbage collection.

14. Click the **Busiest Thread Pools** widget to access the resource dashboard for thread pools.

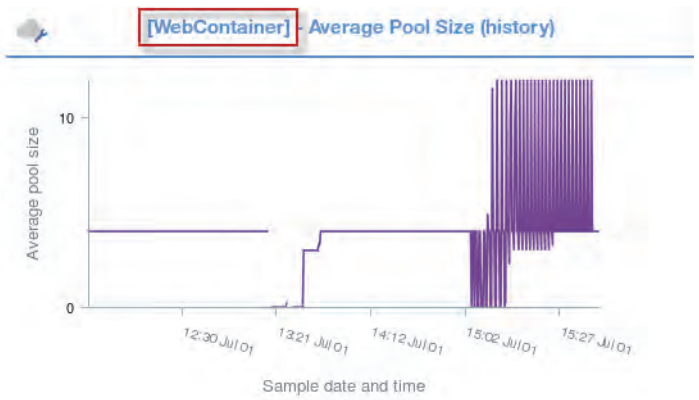


15. Click the header of the **Average Pool Size** column until the highest values are at the top.

16. Select the first item in the sorted **Thread Pool Name** list.

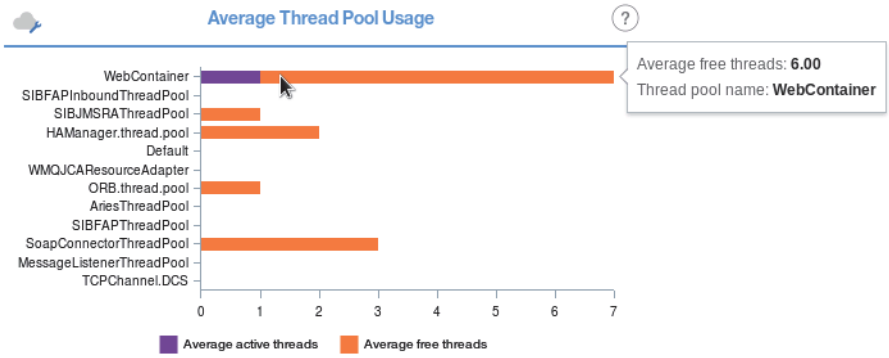


Notice that the added prefix to the header of the **Average Pool Size** widget matches the thread that you selected in the previous step.

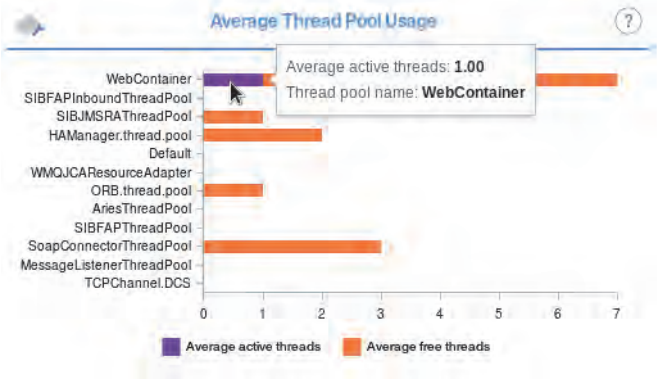


Note: Not all dashboard headers are given the prefix, including **Average Thread Pool Usage**.

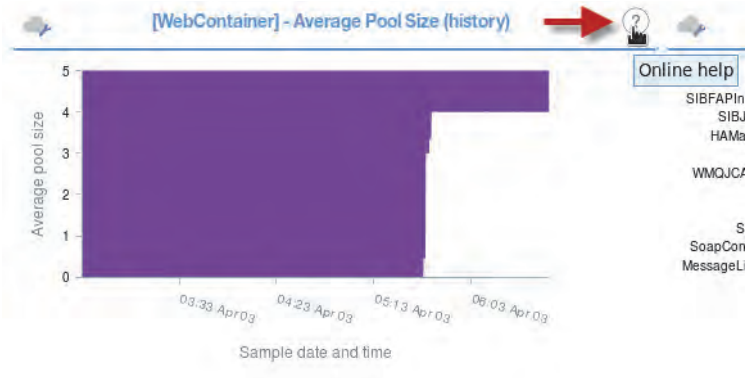
17. Move the mouse pointer over the orange section of the **WebContainer** bar in the **Average Thread Pool Usage** bar chart. Examine the flyover to see the **average free threads** for the selected pool for the current interval.



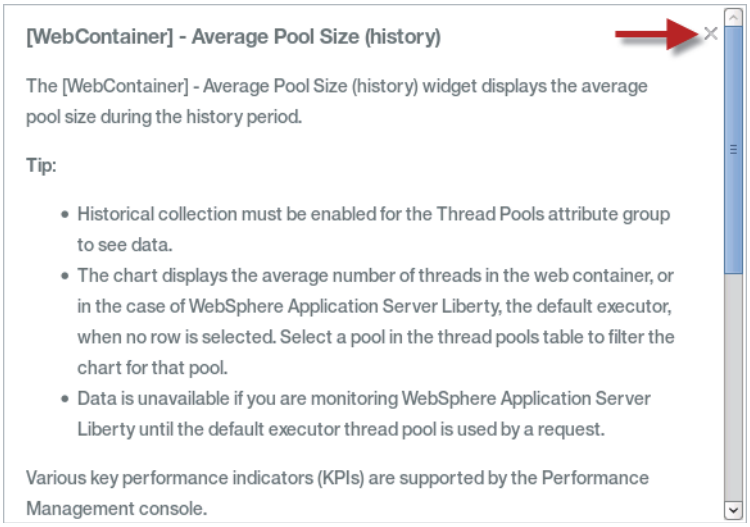
18. Move the mouse pointer over the purple section of the **WebContainer** bar in the **Average Thread Pool Usage** bar chart. Examine the flyover to see the **average active threads** for the selected pool for the current interval.



19. Click the question mark icon at the upper right of the **[WebContainer] Average Pool Size (history)** widget to view attribute details.

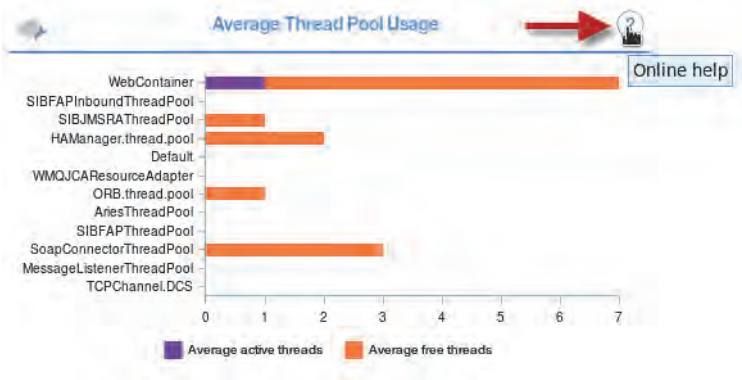


20. Review the help for the **Average Pool Size** widget.

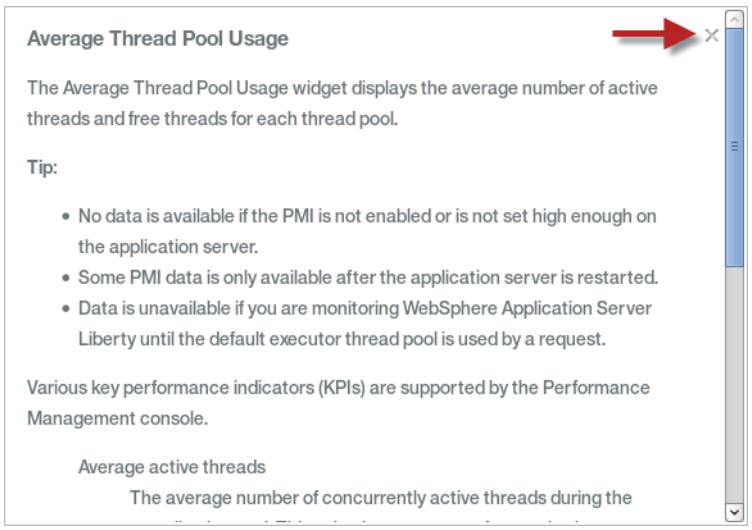


21. Click the **X** to close the window.

22. Click the question mark icon at the upper right of the **Average Thread Pool Usage** widget to view attribute details.



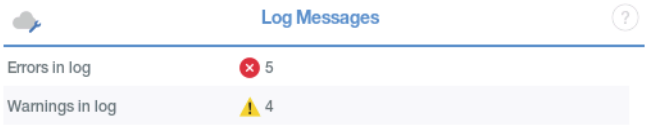
23. Review the help for the **Average Thread Pool Usage** widget.



24. Click the **X** to close the window.

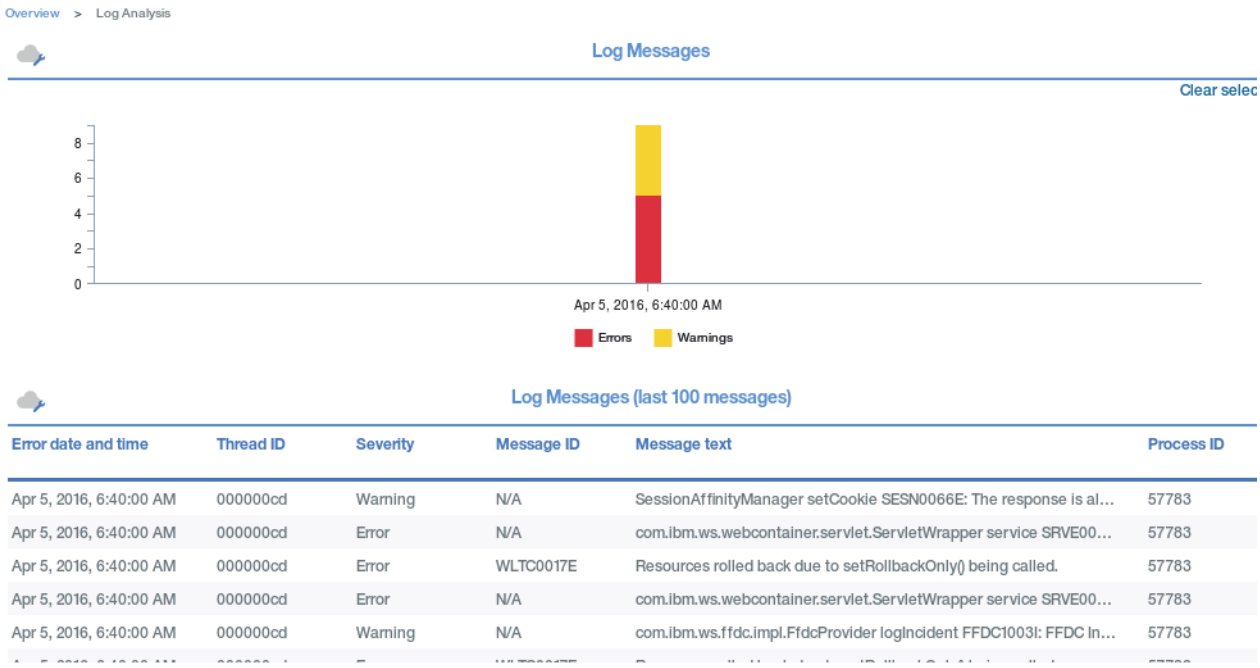
25. Just below the **Status Overview** tab, click the **Overview** link to return to the main WebSphere resource dashboard.

26. Scroll down to the **Log Messages** widget and click it.

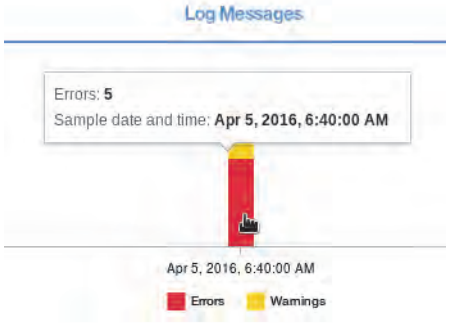


Note: On a lightly loaded system like your lab image, you might not see any messages. This behavior is normal. If you see no messages, proceed to Step 32 on page 21.

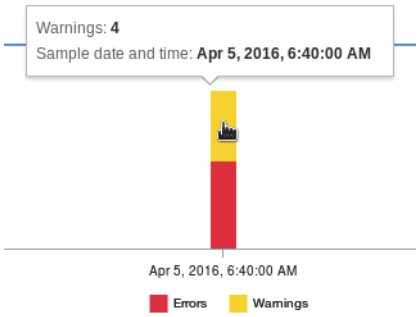
27. Review the **Log Messages** widget.



28. Hover over the red portion of the bar, if present, to see the number of **Errors** in the last 100 log messages.

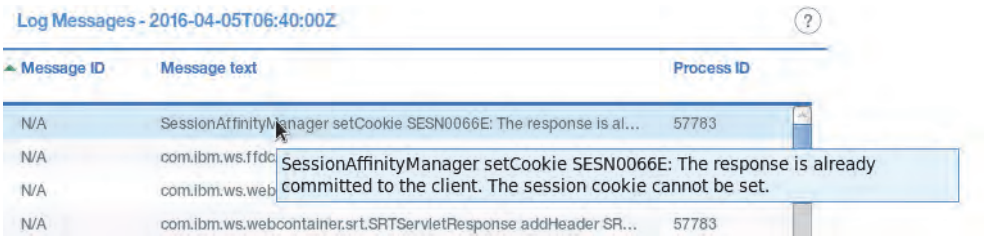


29. Hover over the yellow portion of the bar, if present, to see the number of **Warnings** in the last 100 log messages.

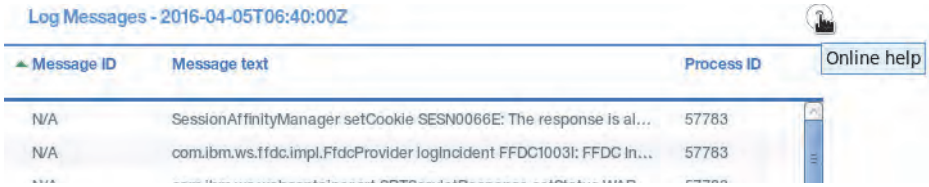


30. Click the bar to access the **Events** dashboard.

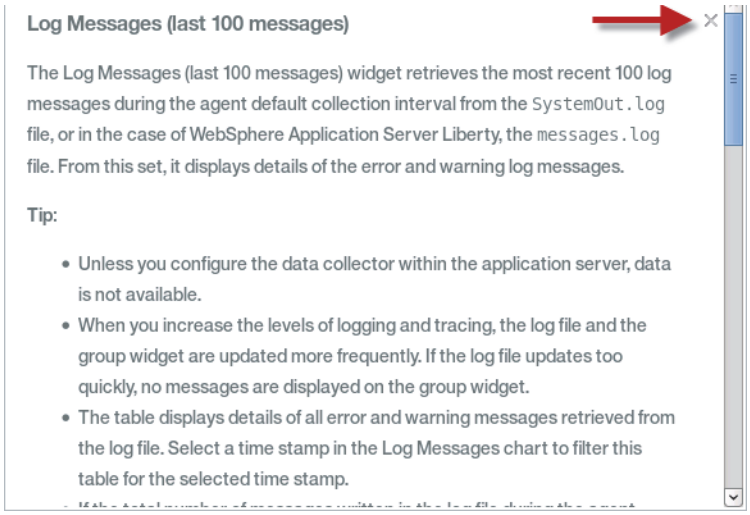
31. Hover over an entry in the **Message text** column to view the full text of a log entry.



32. Click the question mark icon at the upper right of the **Log Messages** widget to view additional information.

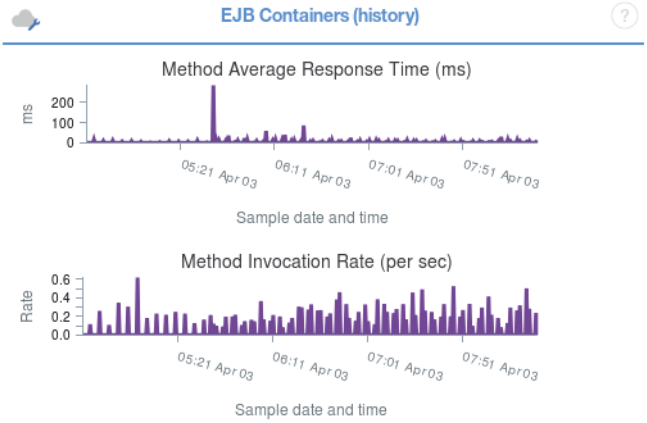


33. Review the help for the **Log Messages** widget.

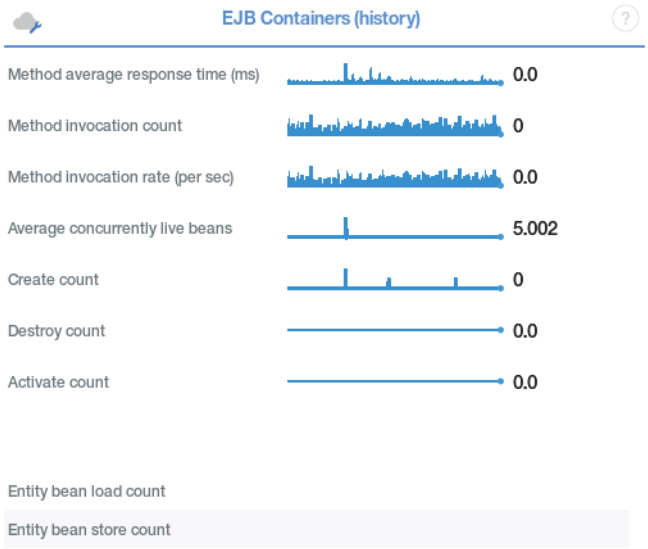


34. Click the **X** to close the window.

35. Click the **EJB Containers (history)** widget to access the resource dashboard.

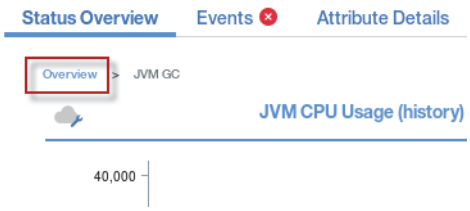


36. Pay particular attention to a widget with the same name, **EJB Containers (history)**, and note the greater level of detail provided.



37. Click the question mark icon for each of the other widgets to view details of the data they provide.

38. Just below the **Status Overview** tab, click the **Overview** link to return to the main WebSphere resource dashboard.

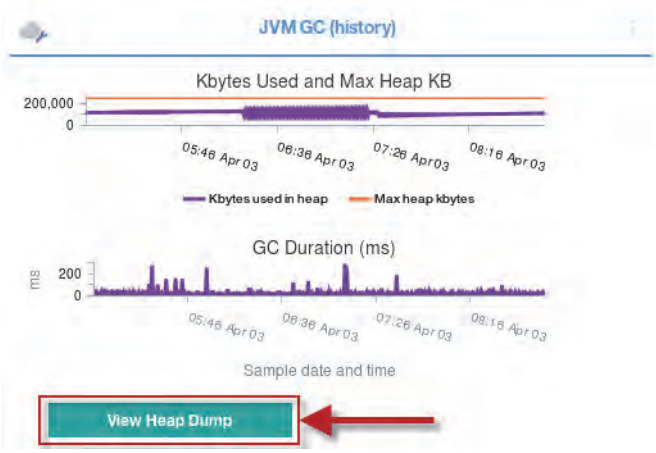


Note: No data might be available for some widgets, such as **Slowest Web Services**. On test systems such as these, that behavior is normal.

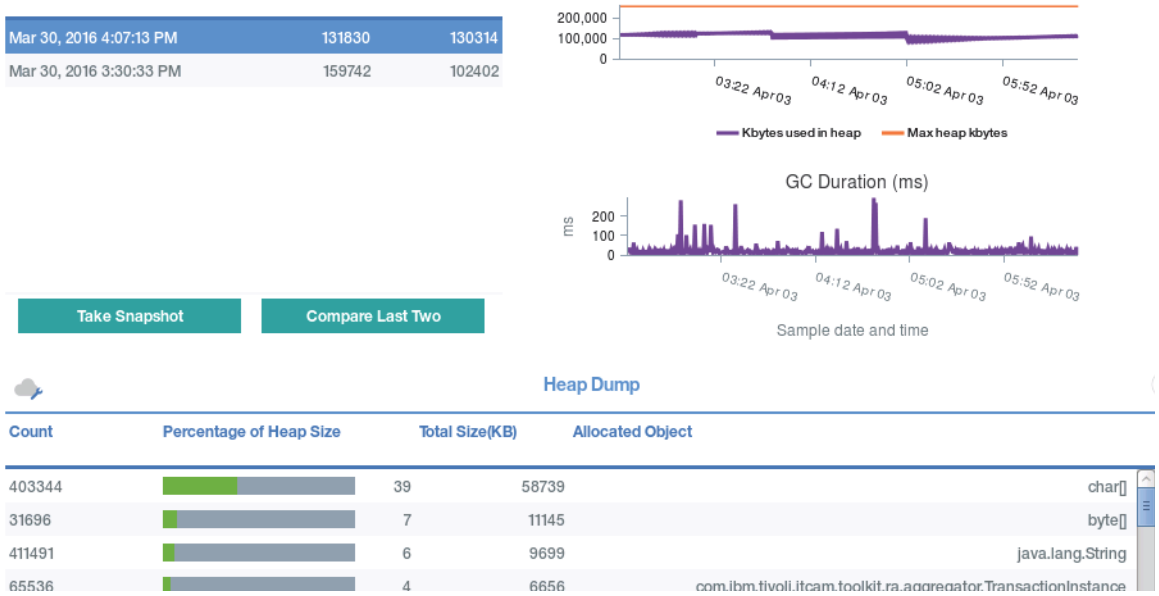
The **Requests with Slowest Response Time**, **In-Flight Request Summary**, and **Heap Dump** functions are covered in the next section.

Exercise 2 Monitoring the heap

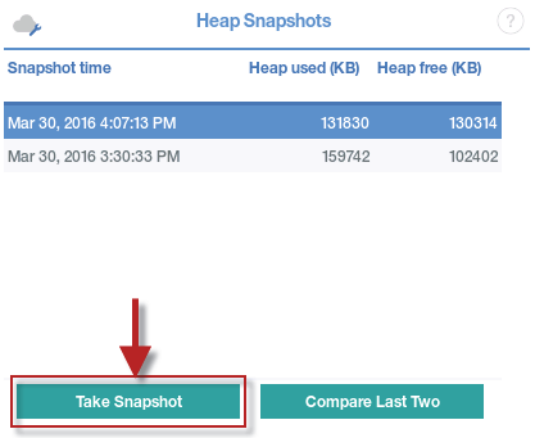
1. Click **View Heap Dump** on the **JVM GC (history)** widget.



2. Scroll down to the **Snapshot** and **Heap Dump** sections of the dashboard.



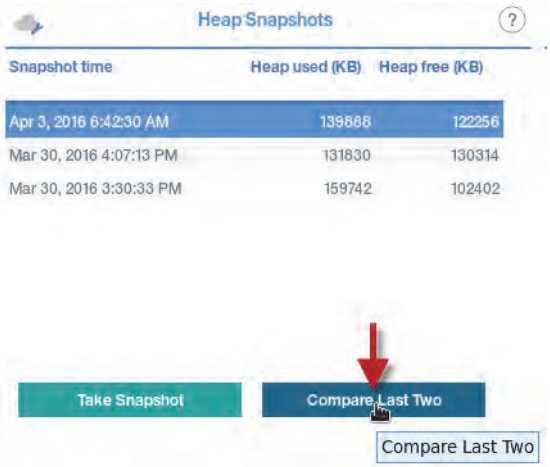
3. Click **Take Snapshot** and wait until a new snapshot opens.



4. Confirm successful creation of the second snapshot.

5. Click **OK** to complete the operation.

6. Click **Compare Last Two**.



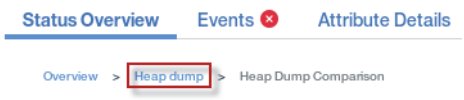
7. Click to the right of the **Percentage of Heap Size** column header until the arrow points down and the heap sorts from the top down.

8. Review the categories of data in the table.

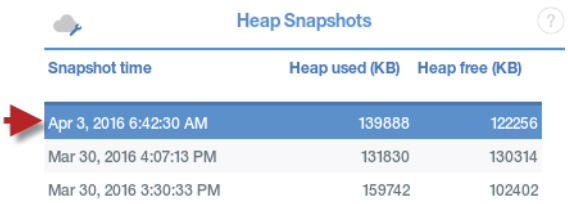
9. Examine the **Delta Count** and **Delta Size (KB)** columns.

Negative numbers, if present, signify a decrease in the size or count for the selected allocated object in the second snapshot.

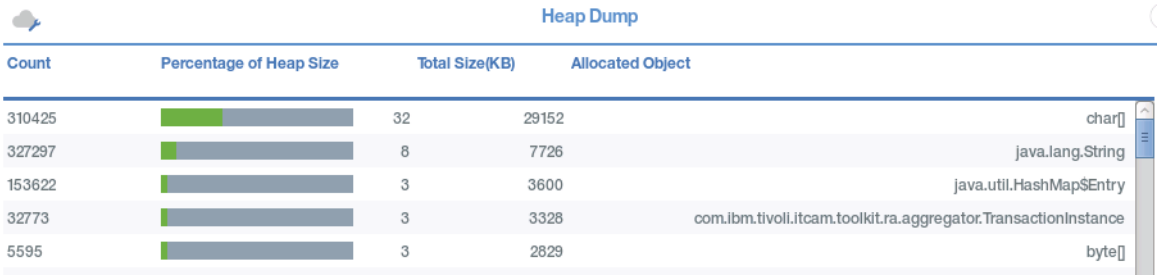
10. Click the **Heap dump** link at the top of the dashboard.



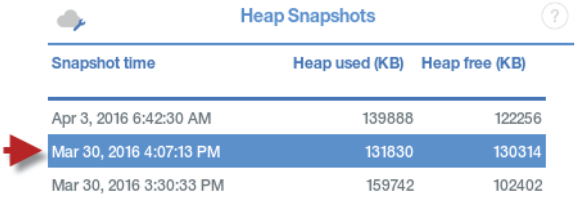
11. Click a row in the **Heap Snapshot** widget.



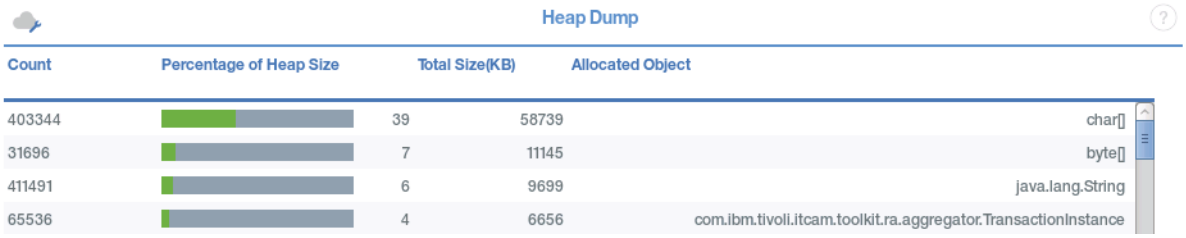
The **Heap Dump** widget at the bottom of the dashboard now shows only the selected heap snapshot.



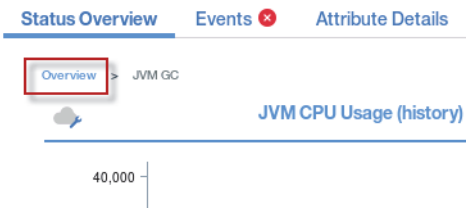
12. Select another snapshot in the **Heap Snapshot** widget.



13. Review the **Heap Dump** widget for the second snapshot.



14. Just below the **Status Overview** tab, click the **Overview** link to return to the main WebSphere resource dashboard.



3 Code-level monitoring

In addition to resource monitoring, IBM Application Diagnostics supports method traces and transaction topologies.

All IBM Application Diagnostics functions, including code-level diagnostic monitoring and application topologies, are available as of this writing for agents that monitor these products:

- WebSphere Application Server
- Microsoft .NET

Code-level diagnostic monitoring is available as of this writing for the agents that monitor these products:

- Node.js
- Ruby

This unit covers transaction monitoring by using the WebSphere monitoring agent.

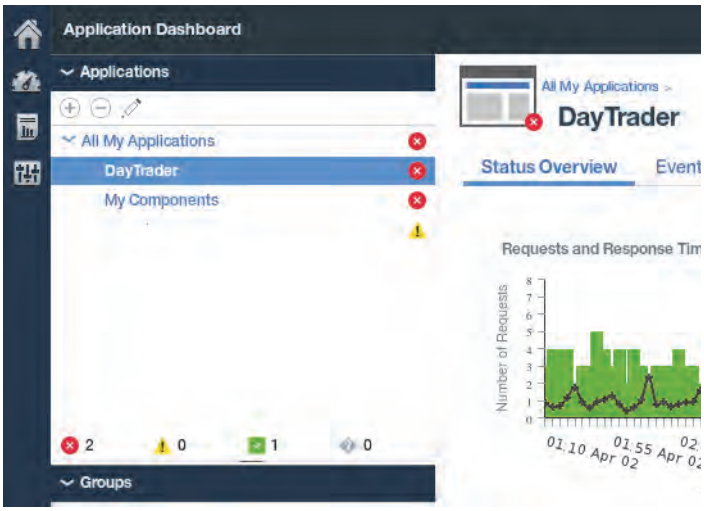
Objectives

After completing all the exercises, you can perform the following tasks:

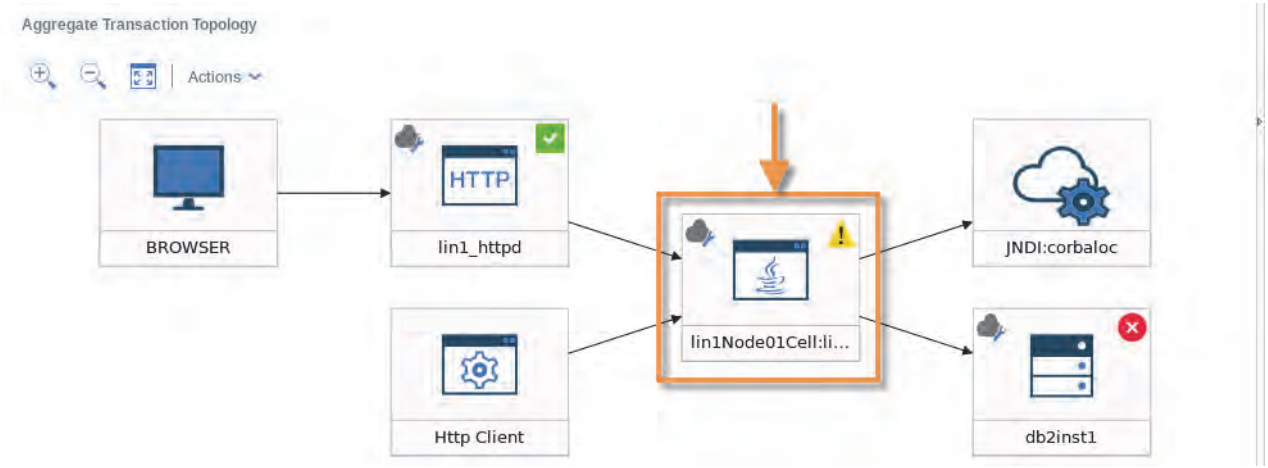
- Analyze request instances, method, and stack traces for WebSphere Application Server
- Analyze transaction topologies for WebSphere Application Server

Exercise 1 Accessing code-level data for WebSphere

1. Return to the top-level **Status Overview** dashboard for the **DayTrader** application.

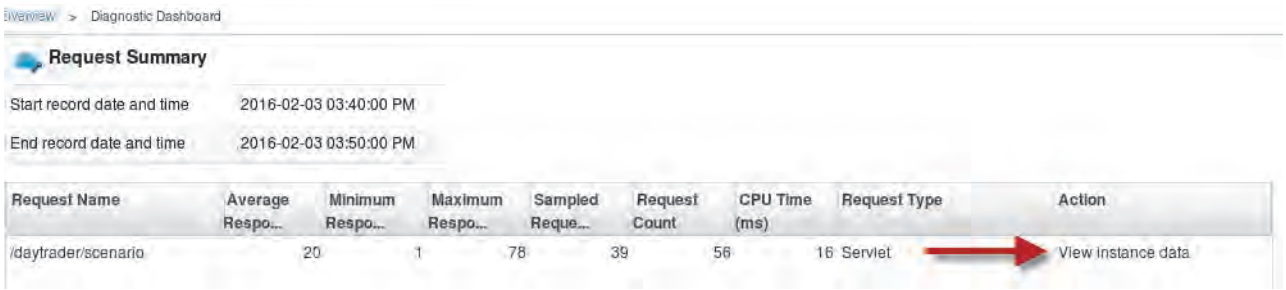


2. Double-click the WebSphere topology node.

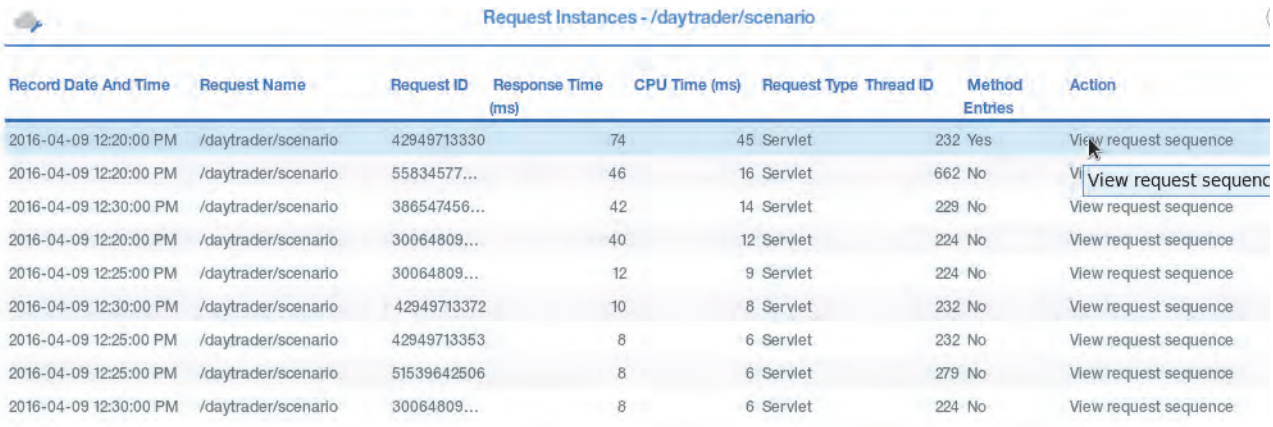


3. Click **Diagnose**.

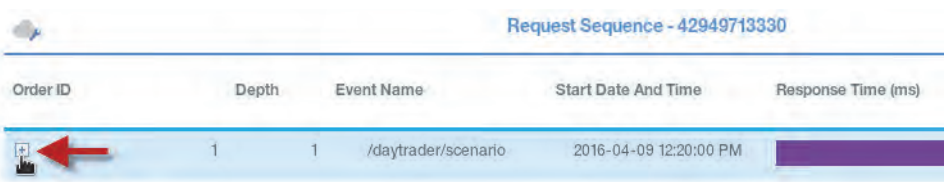
4. Click **View Instance data** for the transaction instance in the list with the slowest **Average Response Time**.



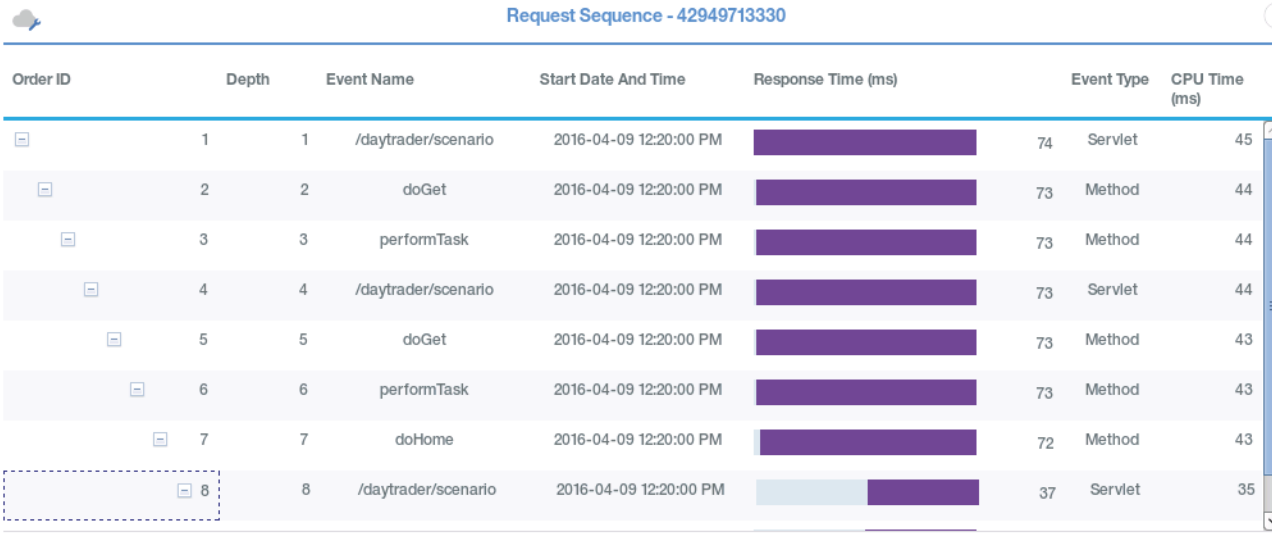
5. Click **View request sequence** for the request instance with the slowest **Response Time**.



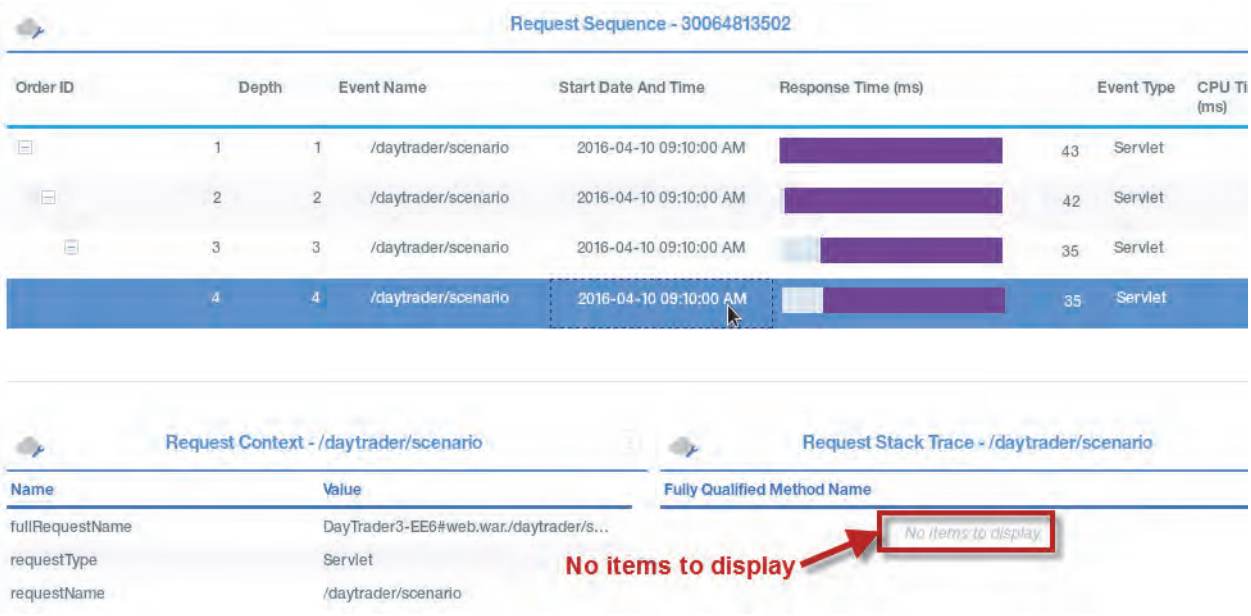
6. Click the plus (**+**) sign to expand the request.



7. Continue clicking any subsequent plus (**+**) signs until the request is fully expanded.



8. Click through the request events, and note any events that have an associated **Stack Trace, Request Context**, or both.

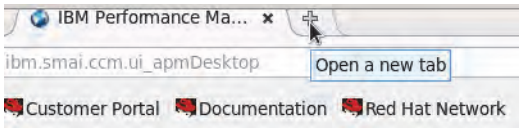


Note: Your requests might not have data in one of both of the request widgets.

Exercise 2 Generating more traffic

If none of the **daytrader/scenario** transactions produces a stack trace, generate more transactions by stopping and restarting all applications in the WebSphere administrator console.

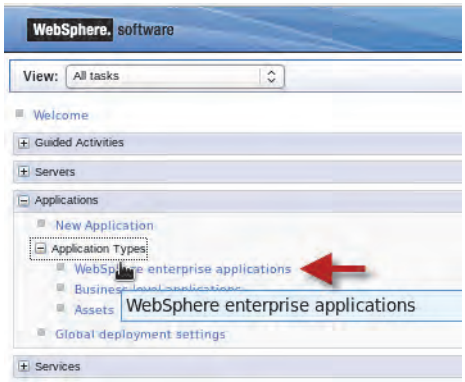
1. Click the plus (**+**) sign to open a new tab in your browser.



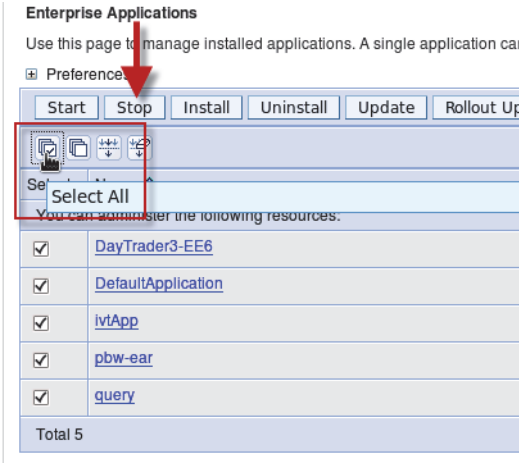
2. Enter **lin1:9060/ibm/console** in the address bar of your browser and press **Enter**.

3. Enter the user ID **wasadmin** and click **Log in**.

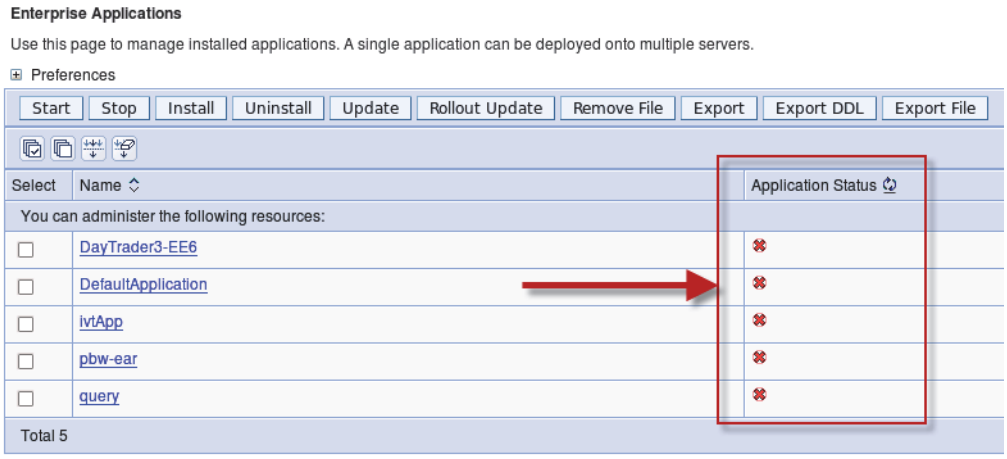
4. Navigate to **Applications > Application Types > WebSphere enterprise applications**.



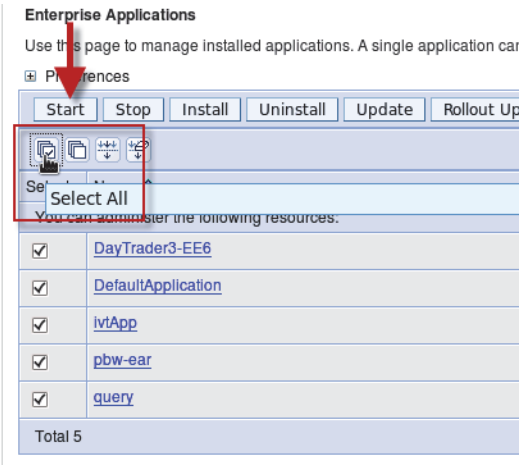
5. Click the **Select All** icon; click **Stop**.



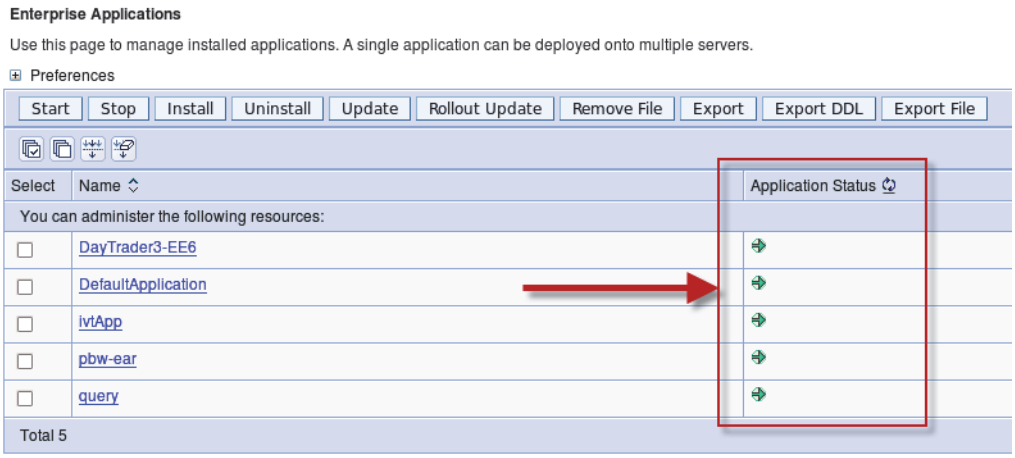
6. Confirm that the applications were stopped successfully.



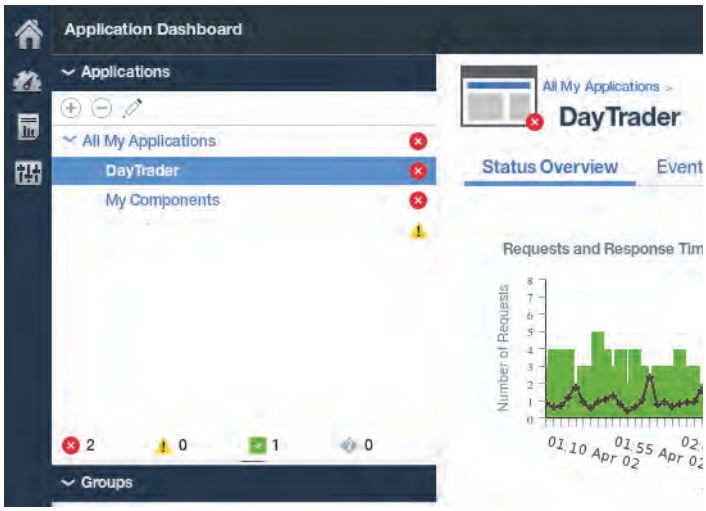
7. Click the **Select All** icon; click **Start**.



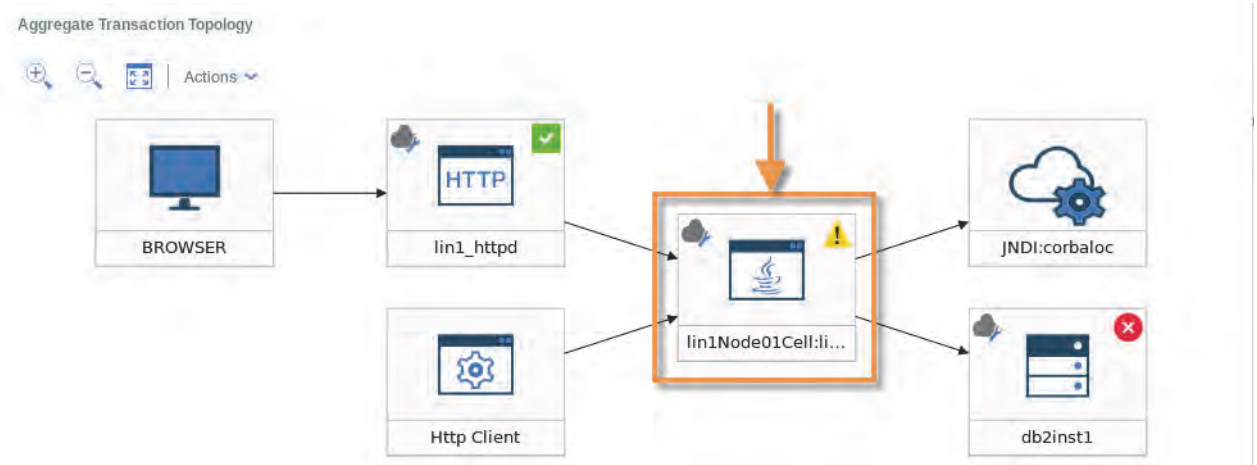
8. Confirm that the applications were started successfully.



9. Return to the **Performance Management** console and select **DayTrader** in the application list.



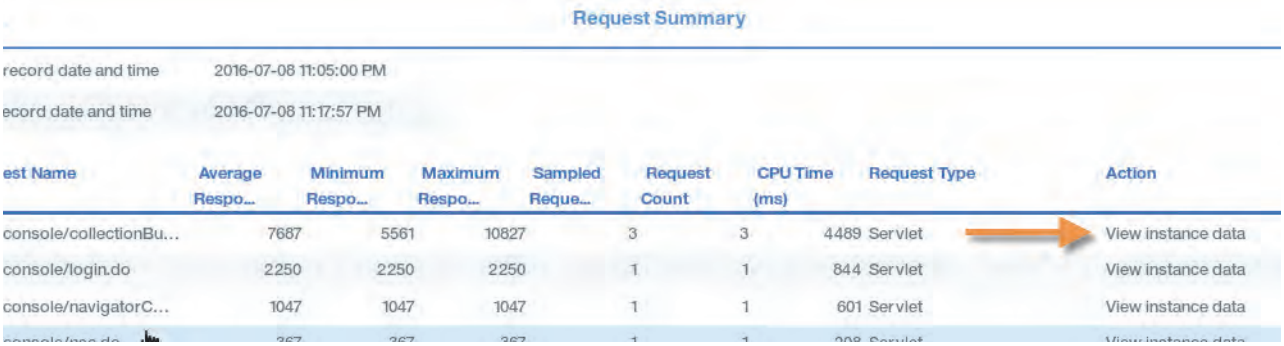
10. Double-click the WebSphere topology node.



11. Review the list of **Requests with Slowest Response time**, noting especially requests that contain the string **/ibm/console/.**

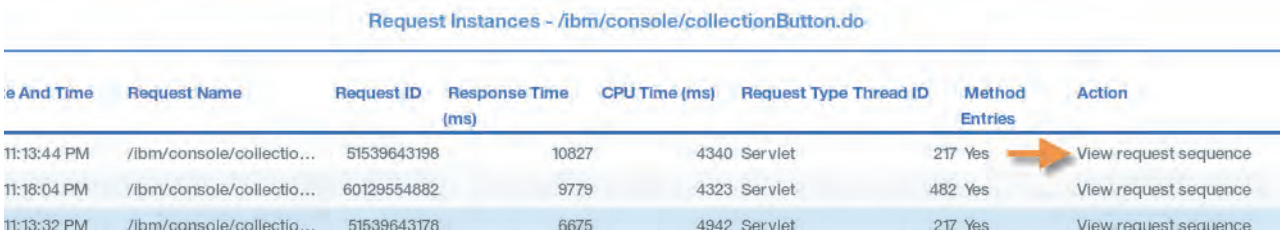
12. Click **Diagnose** at the lower left of the **Requests with Slowest Response time** widget.

13. Click **View Instance data** at the right of the worst performing transaction in the **Request Summary** widget.

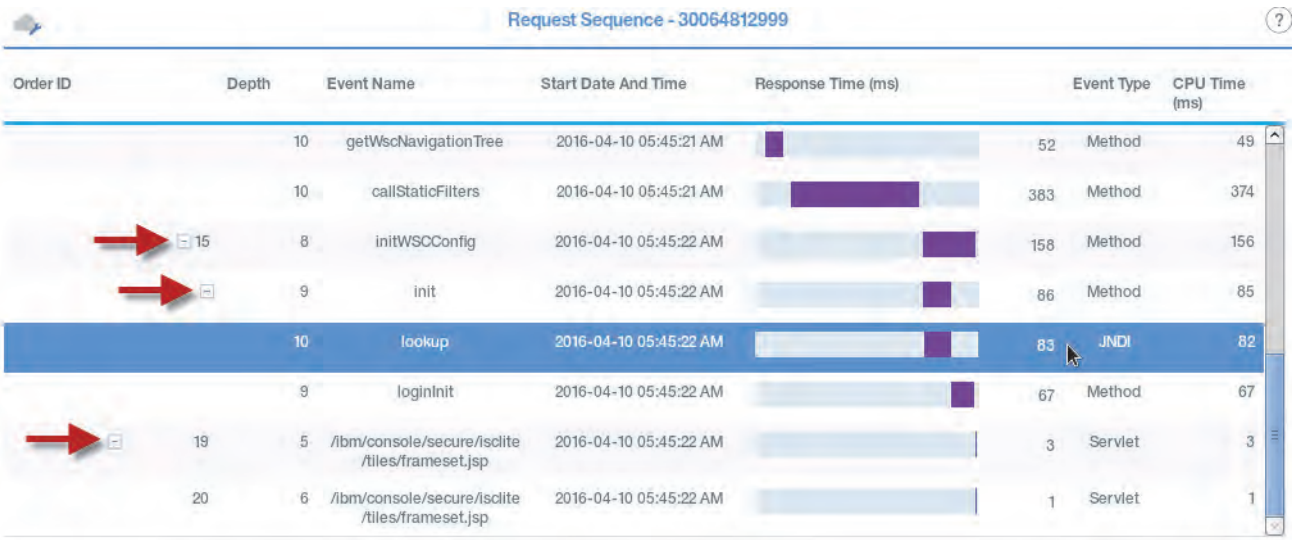


Note: The details of your list of transactions might be different.

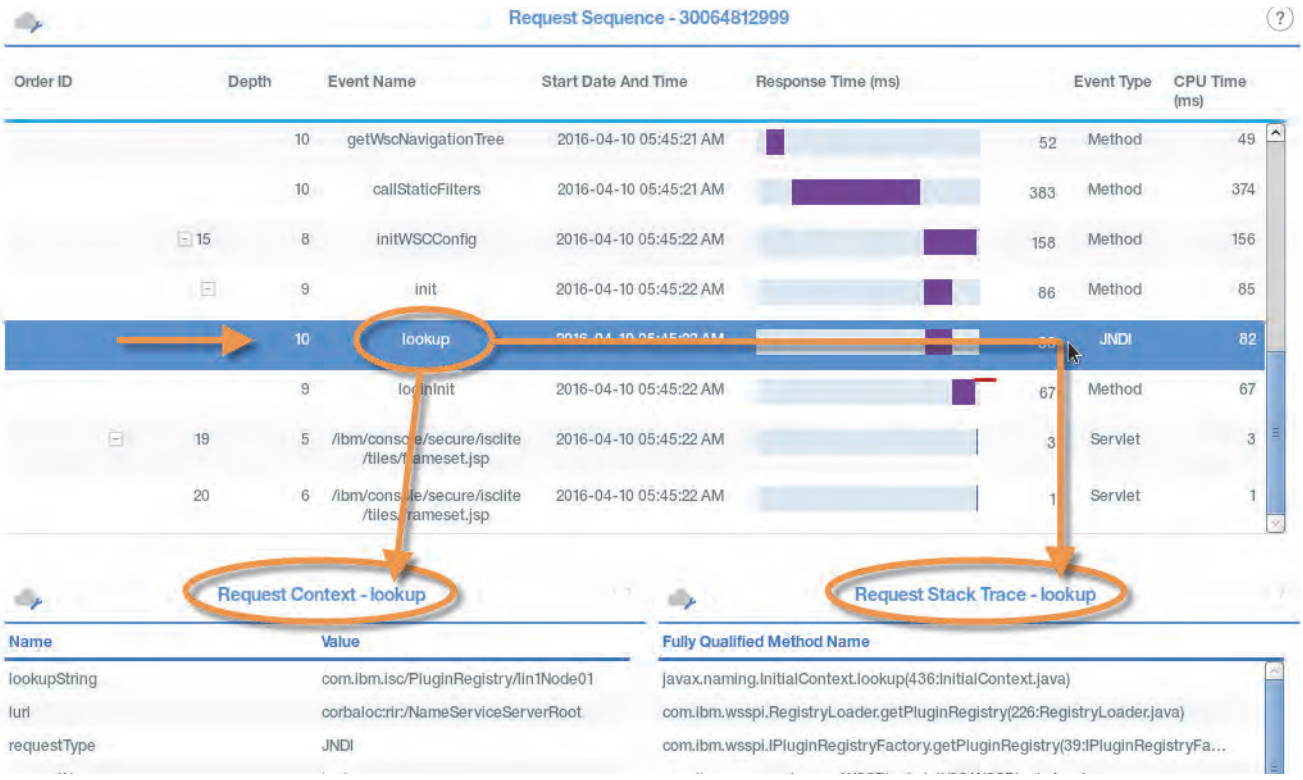
14. Click **View request sequence** at the right of the worst performing transaction in the **Request Instances** widget.



15. Click any plus (**+**) signs to expand the request.



16. Click through the request events, and note any events that have an associated **Request Context**, **Stack Trace**, **Method Summary**, or a combination of the three.



17. Click the question mark icon of the **Request Stack Trace** widget to access online help for details of how to interpret the data.

4 Transaction tracking

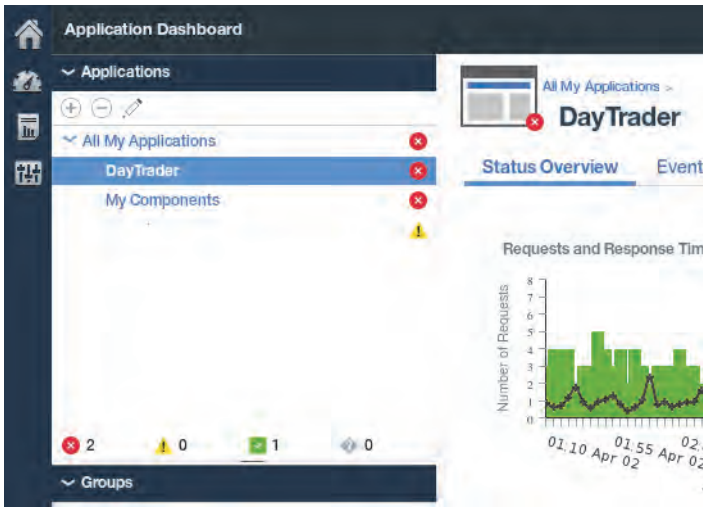
In addition to resource monitoring and code-level diagnosis, IBM Application Performance Management Advanced Diagnostics supports transaction tracking.

Objectives

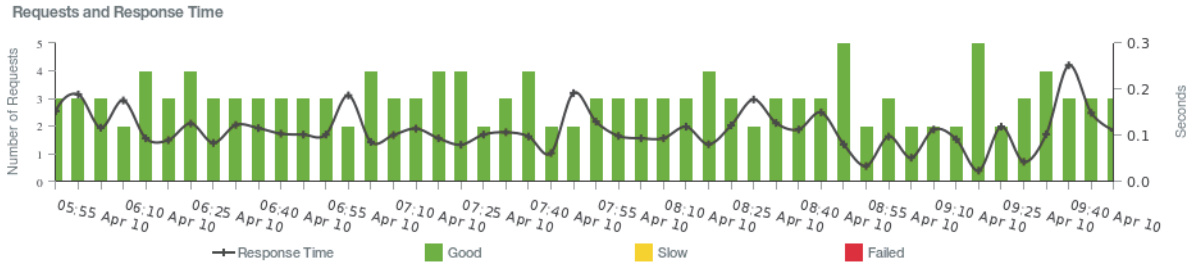
After completing all the exercises, you can analyze transaction topologies for WebSphere Application Server.

Exercise 1 Exploring aggregate topologies

1. Return to the **Performance Management** console and select **DayTrader** in the application list.

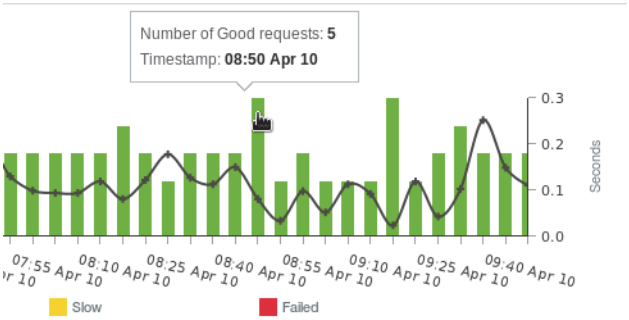


2. Review the chart for top-level **Requests and Response Time** widget.

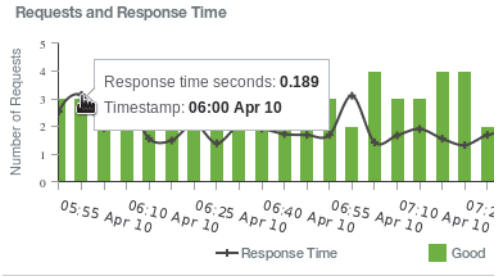


The bars indicate the number of requests for a specific collection period. The three colors are for Good, Slow, and Failed requests. The single line at this level graphs response time.

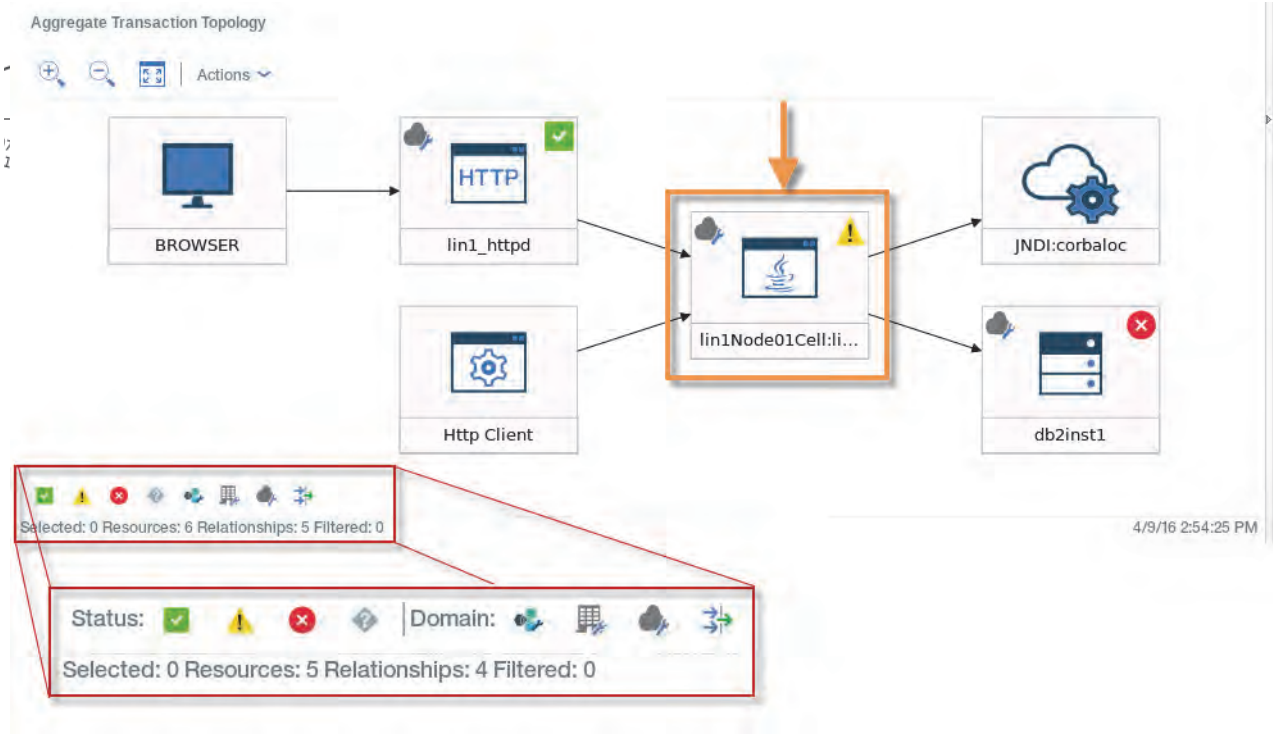
3. Position the cursor over one of the bars, and note the request and time information.



4. Position the cursor over one of the nodes on the response timeline to see the flyover. The node is directly above or on top of the corresponding bar.



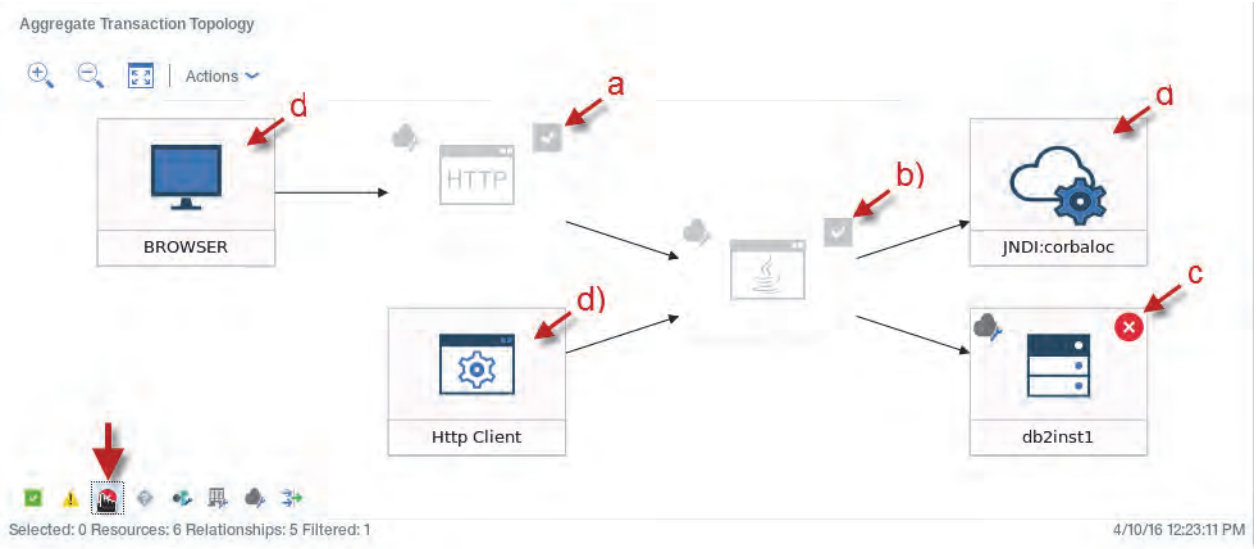
5. Review this sample **Aggregate Transaction Topology**, including the icons at the lower left.



Note: Your topology might look different.

If you click the Critical icon in that example, you see the topology at Step 6 on page 40.

6. Note the changes in the **Aggregate Transaction Topology** when the critical icon is selected.



   a. The **lin1 HTTP** node previously had a normal status, which was indicated by a white check mark in a green box (see the screen capture in Step 5 on page 39). The node is now disabled **for display purposes only**.

   b. The **lin1Node01Cell** previously had a normal status, which is indicated by a white check mark in a green box, and is now disabled **for display purposes only**.

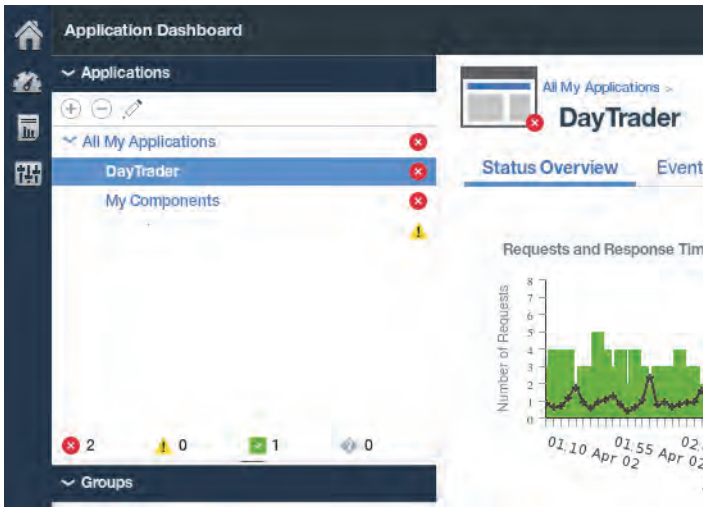
   c. The **db2inst1** node was in a critical state and so is still displayed normally.

   d. The **BROWSER** and **JNDI:corbaloc** nodes are unmonitored. Because they have no assigned status, those nodes are unaffected by the filter.

7. Click the **Critical** node again to turn off the filter, returning the topology to the state shown at Step 5 on page 39.

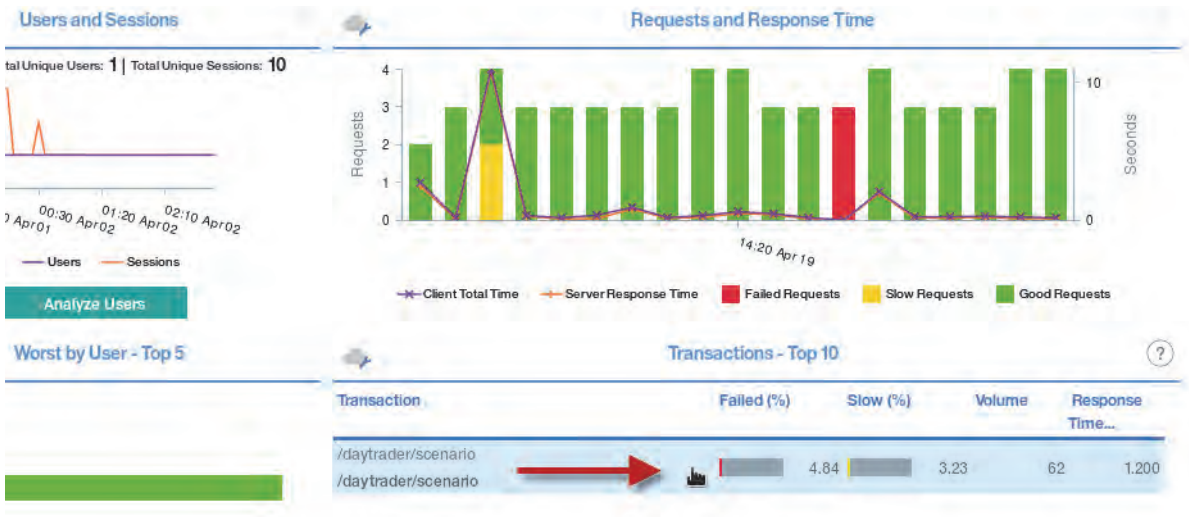
Exercise 2 Exploring transaction instance topologies

1. Return to the top-level **Status Overview** dashboard for the **DayTrader** application.

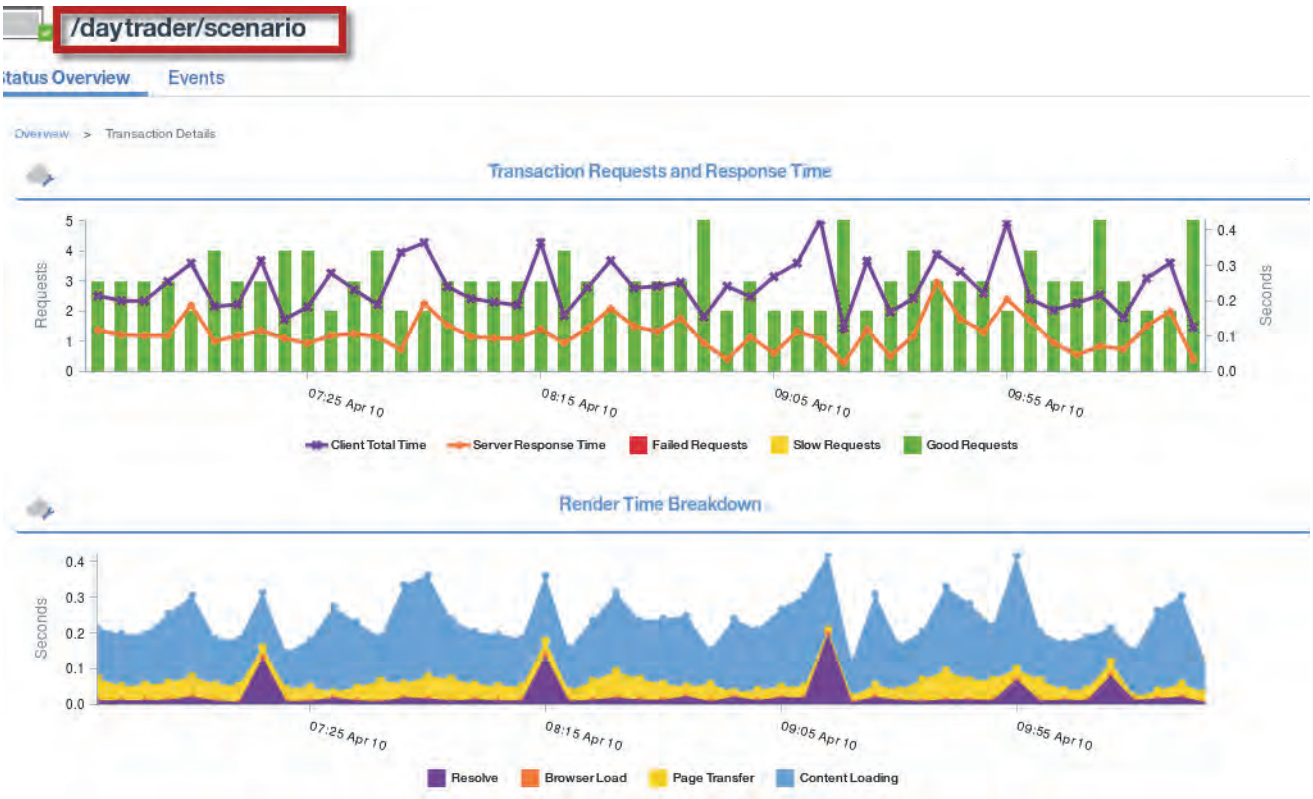


2. Click anywhere on the **Requests and Response Time** widget.

3. Locate and click the **/daytrader/scenario** transaction in the **Transactions Top 10** list.

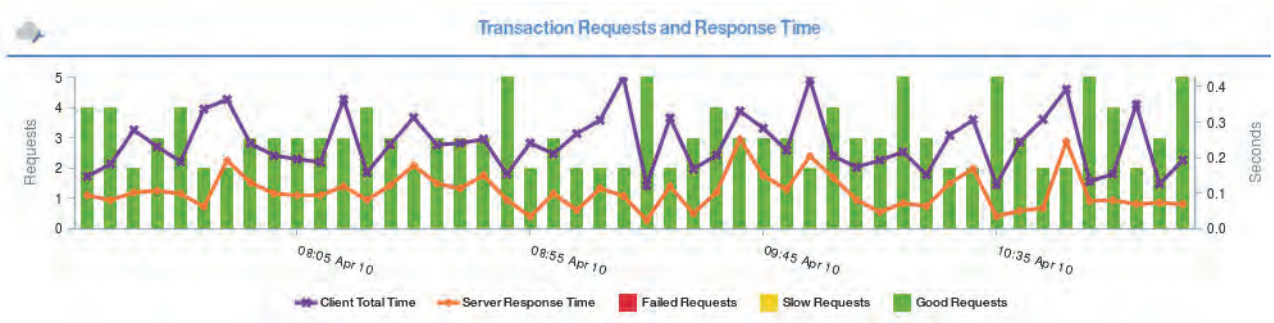


4. Review the two graphs at the top of the **/daytrader/scenario** dashboard.



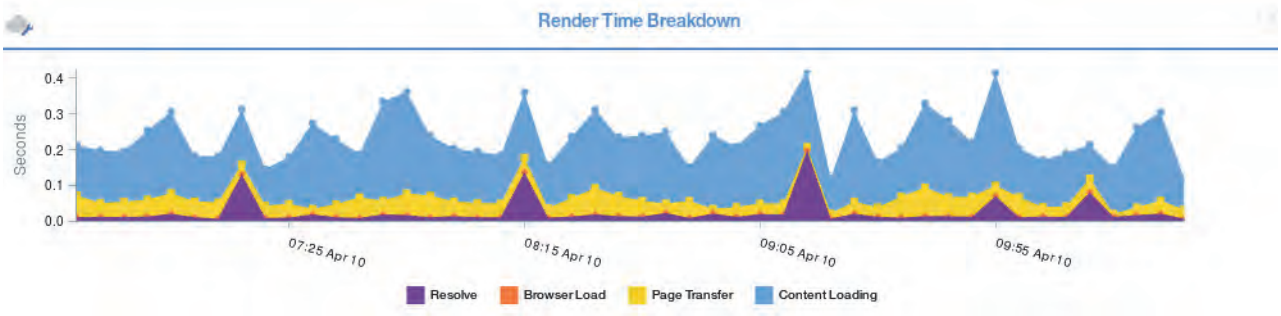
The **Transaction Requests and Response Time** graph is similar to the graph at Step 2 on page 41.

5. Notice the added detail in the second-level **Requests and Response Time** chart, with a response time breakdown for **Client Total Time** and **Server Response Time**.



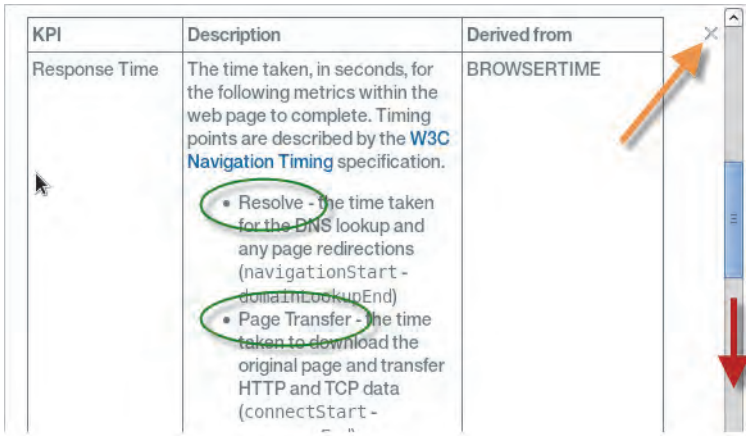
Note: The **Client Total Time** in this second-level graph corresponds to the **Response time**line in the top-level graph.

6. Review the four elements of the **Response Time Breakdown** graph.



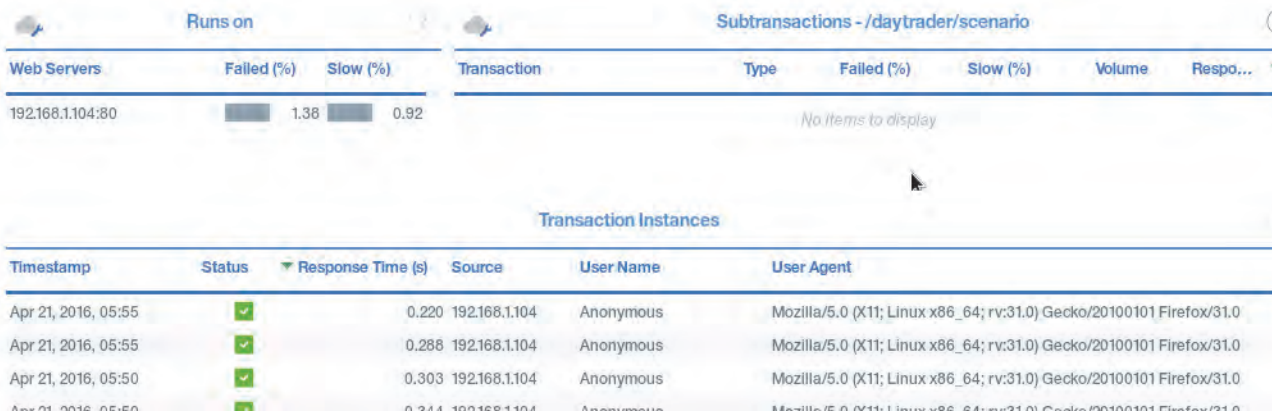
7. Click the question mark icon of the **Render Time Breakdown** widget for details of how to interpret the data.

8. Scroll down in the Help dialog to the definitions for the four render time elements.



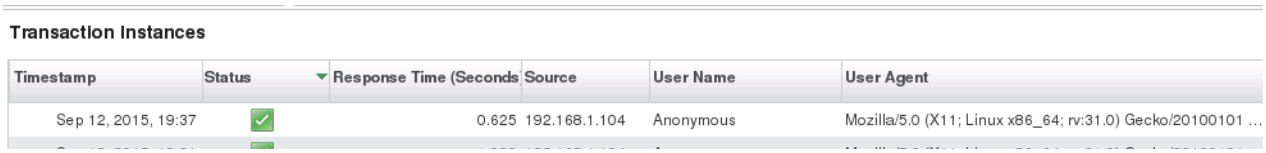
9. Click the X to close the help dialog.

10. Scroll down and review the lower portion of the **End User Transactions** dashboard.

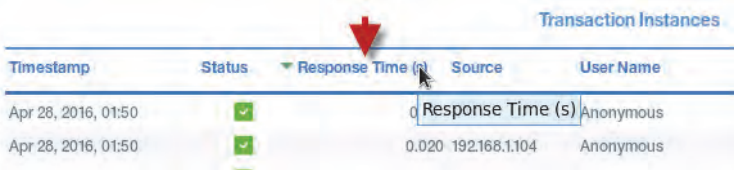


Note: The lack of subtransactions in this case is normal.

11. Scroll down in the same dashboard to the **Transaction Instances** widget.

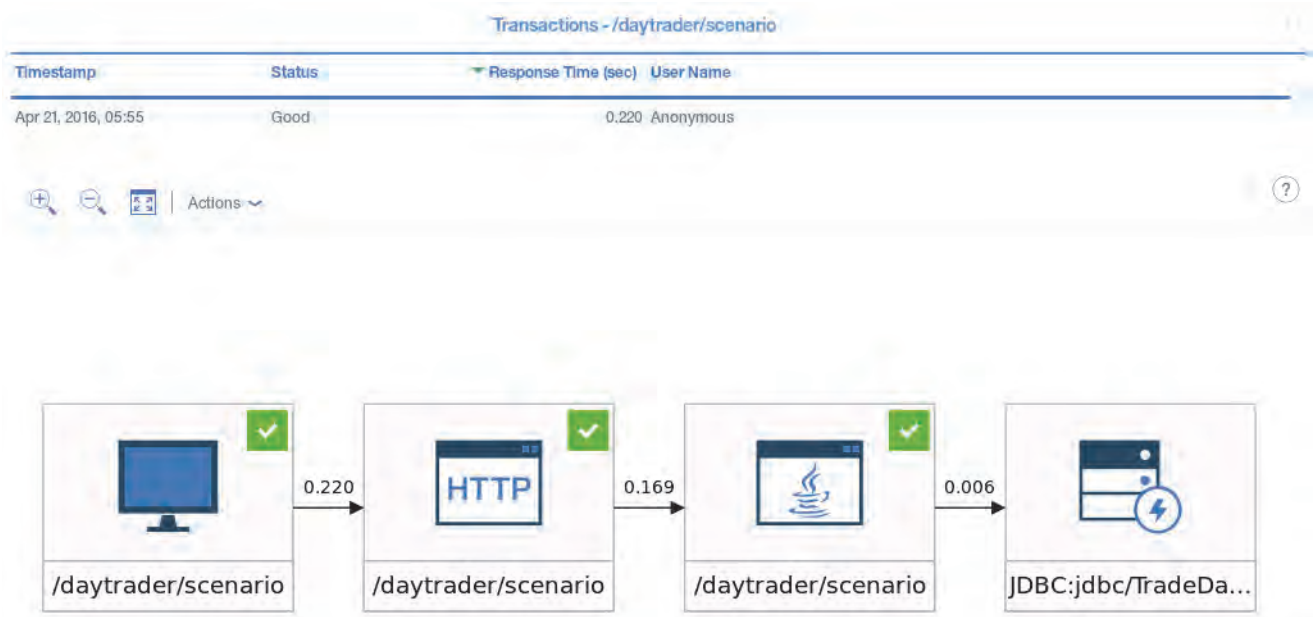


12. Click the **Response Time** column header to bring the longest response time to the top. You might have to click the header more than multiple times.



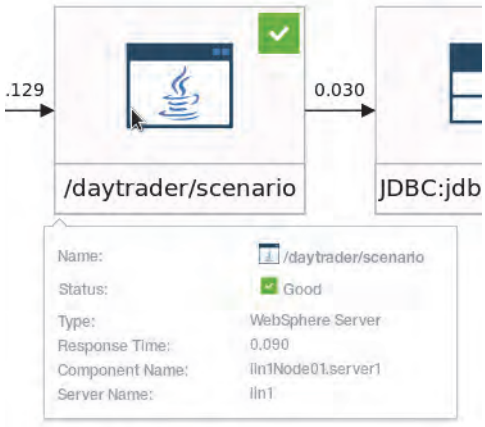
13. Click the top transaction in the list.

14. Review the topology.



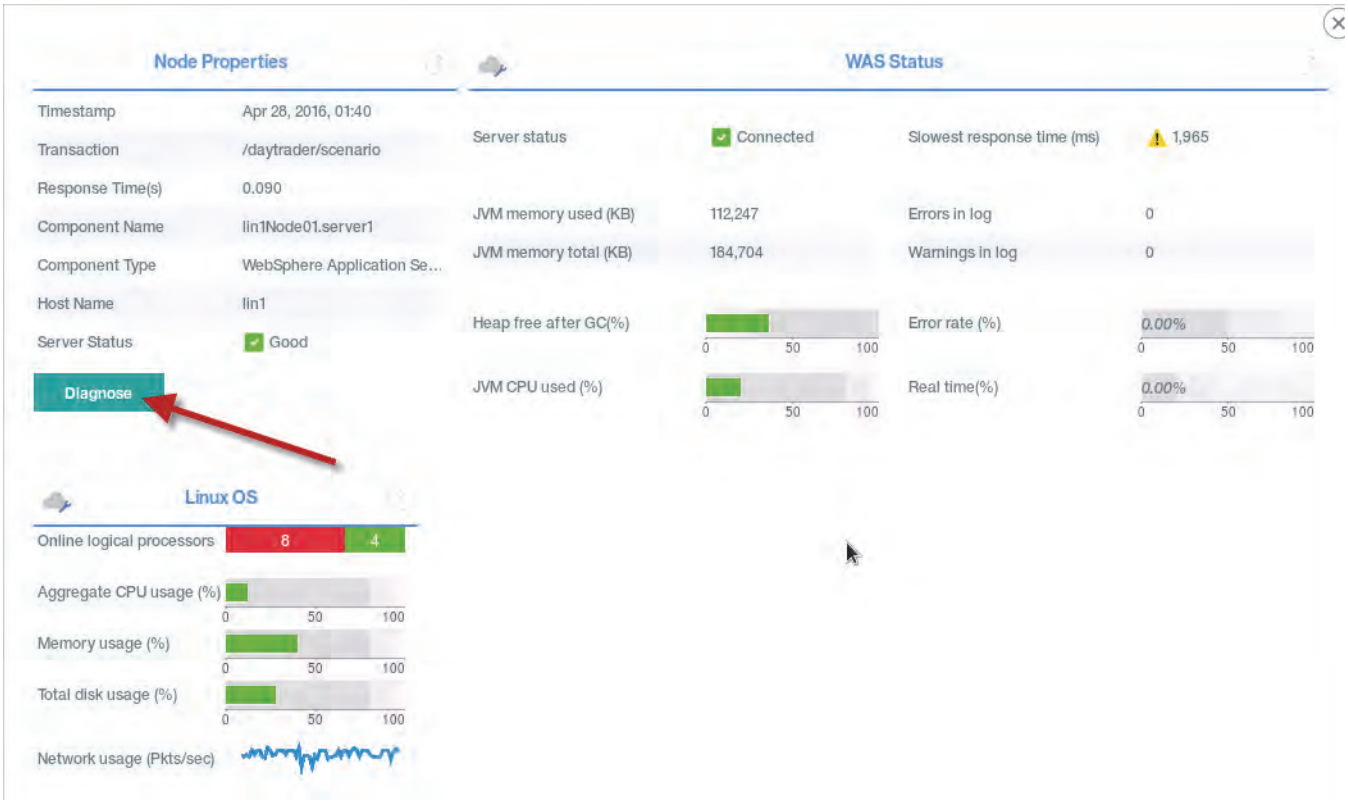
Note: The node at the right represents an uninstrumented external service. The details of your topology might be different.

15. Hover over a topology node to view node-specific details.

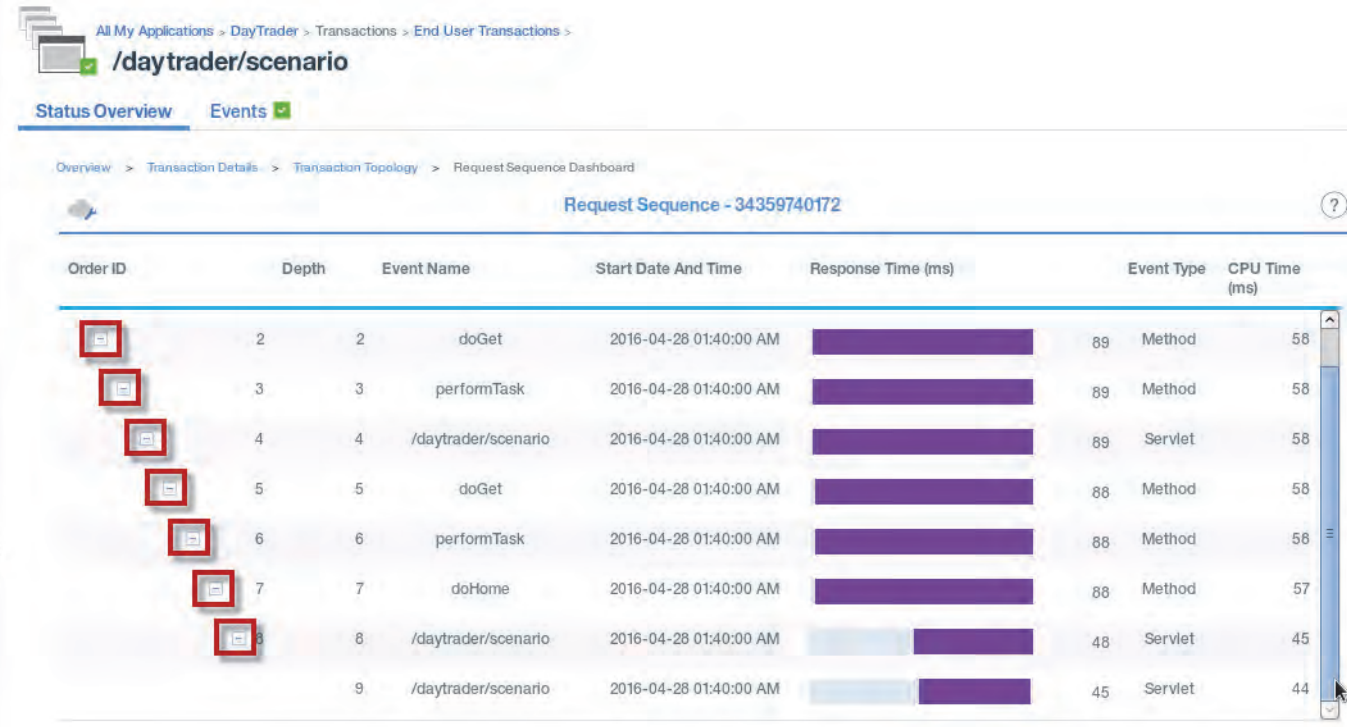


16. Click the **/daytrader/scenario** node.

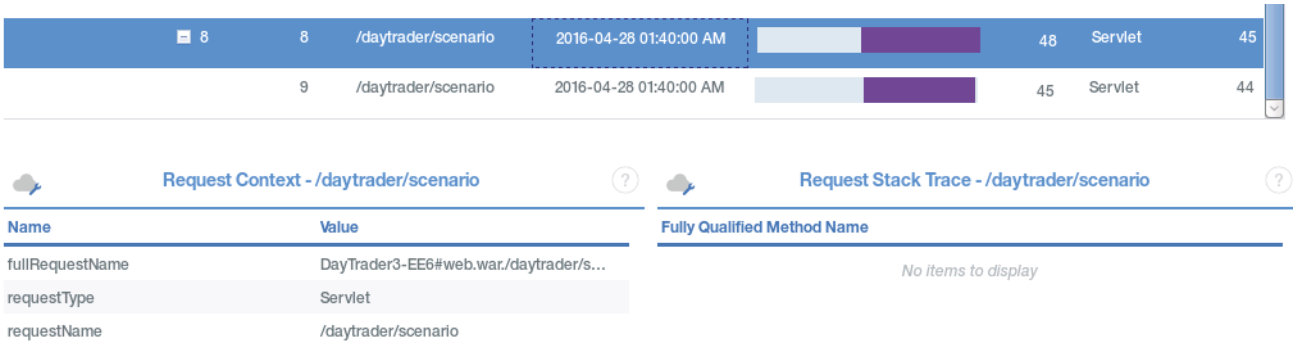
17. Click **Diagnose** on the **Node Properties** widget.



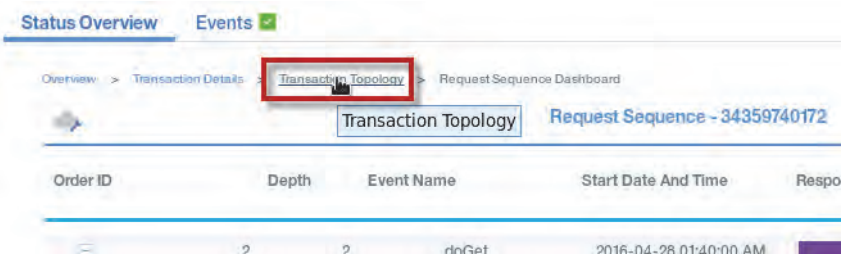
18. Click all plus (**+**) signs to fully expand the **Request Sequence**.



19. Double-click individual events to see whether they have an associated **Request Context** or **Request Stack Trace**d.

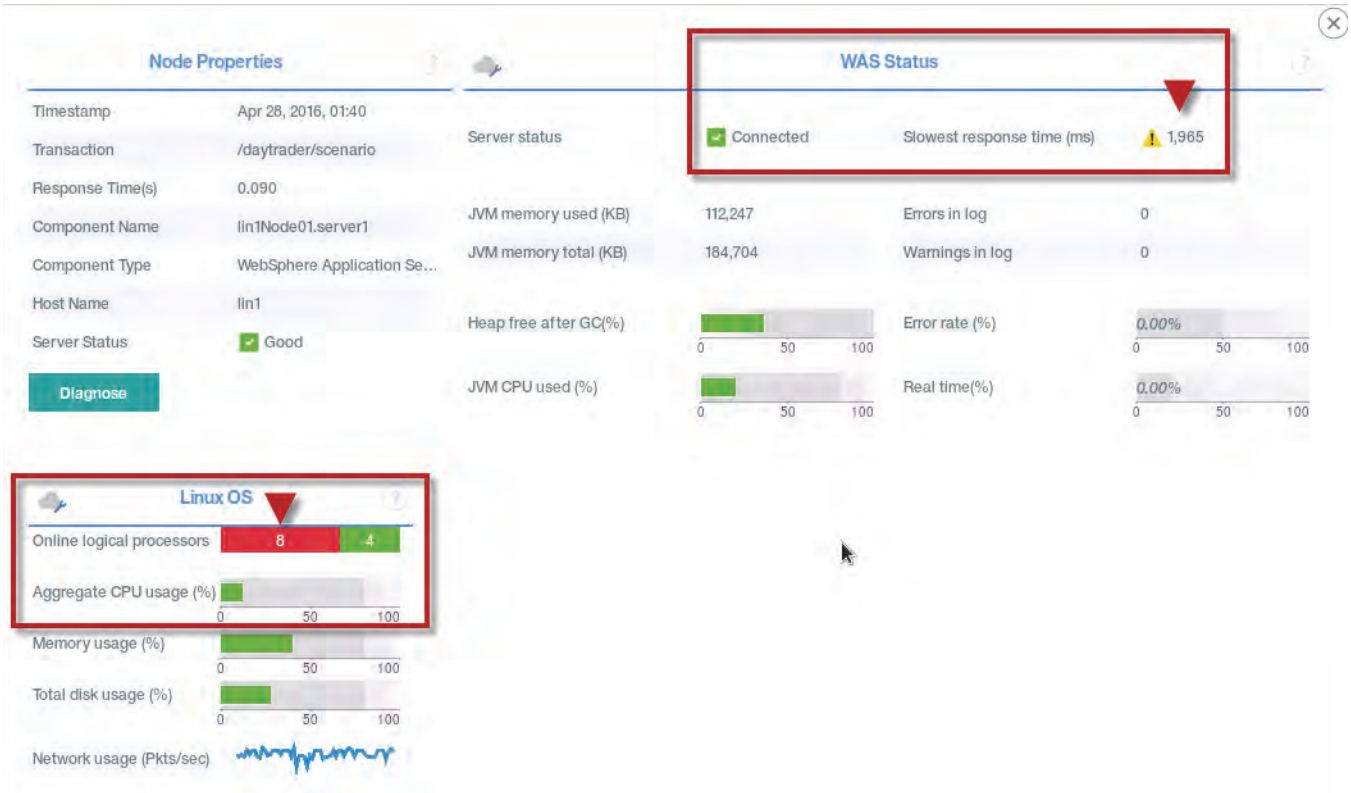


20. Click the **Transaction Topology** link at the top of the dashboard.



21. Click the **/daytrader/scenario** node.s.

22. Consider drilling down into the **WAS Status** and **Linux OS** nodes for further investigation.



5 Synthetic transaction and user monitoring

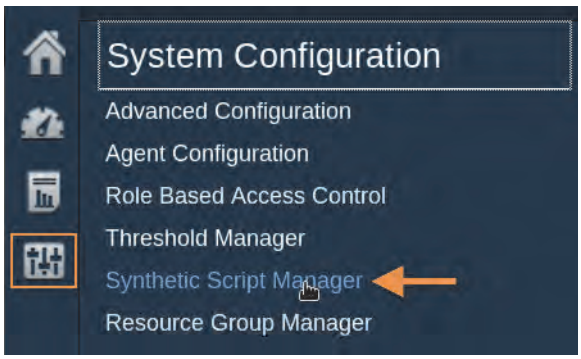
In these exercises, you create a synthetic transaction for Playback and monitoring. By completing these tasks, you enable periodic monitoring of the website that is accessed by the script.

Objectives

After completing all exercises, you can create and monitor synthetic transactions.

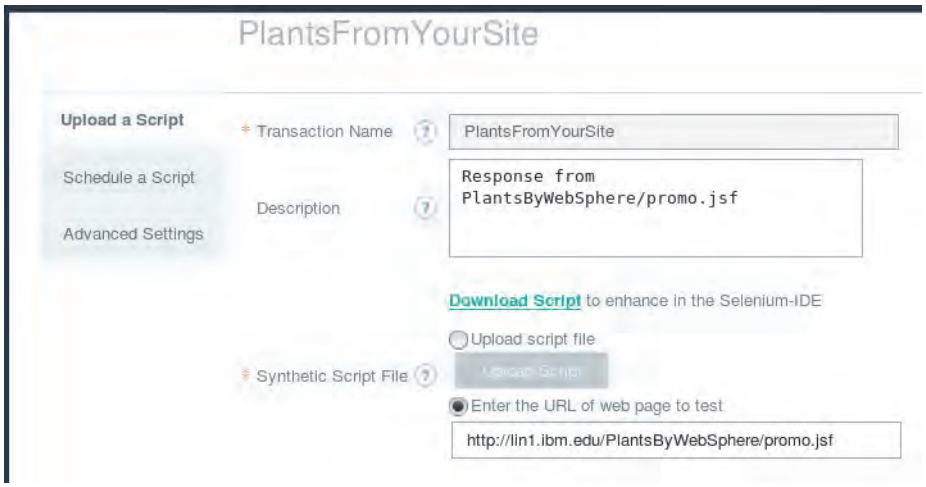
Exercise 1 Configuring a synthetic transaction

1. In the Performance Management console, hover over the sliders icon to bring up the **System Configuration** menu; then, select **Synthetic Script Manager**.



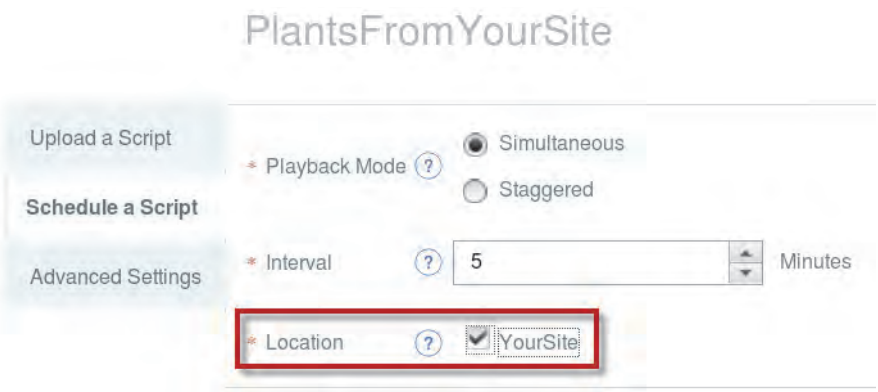
2. Click the plus (**+**) sign to create a new transaction.

3. On the **Upload a Script** tab of the Synthetic Script Editor, enter the following parameters:
   a. Transaction Name: **PlantsFromYourSite**
   b. Description: **Response from PlantsByWebSphere/promo.jsf**.
   c. Select URL option and enter **http://lin1.ibm.edu/PlantsByWebSphere/promo.jsf**



4. Click the **Schedule a Script** tab and set the following Playback schedule and location.
   a. Playback Mode: **Simultaneous**
   b. Interval: 5 minutes

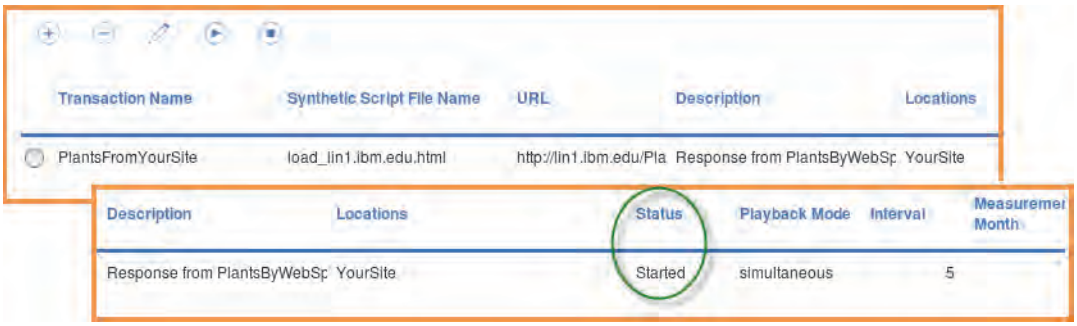
c. Location: **YourSite**



5. Click **Save Transaction** to finish creating the transaction.
6. Click **OK** to close the confirmation window.

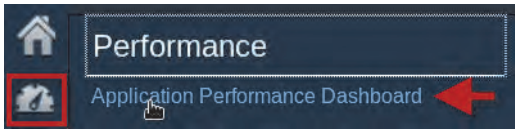
You are returned to the Synthetic Script Manager.

7. Confirm that the transaction started.

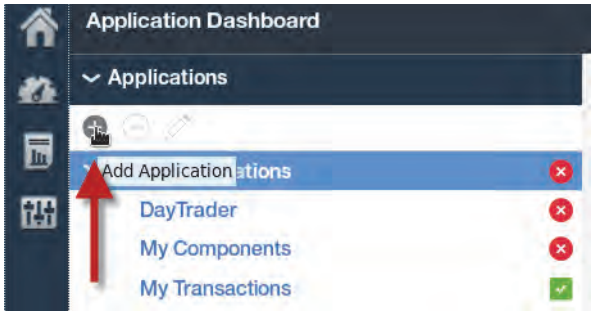


Exercise 2 Creating a synthetic application

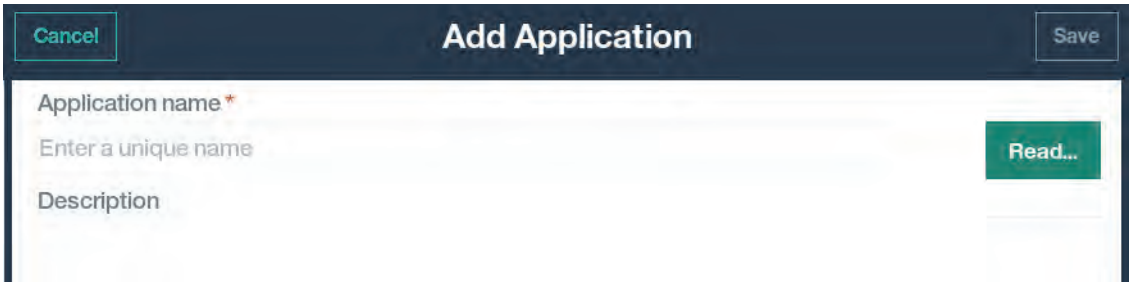
1. Return to the **Application Performance Dashboard**.



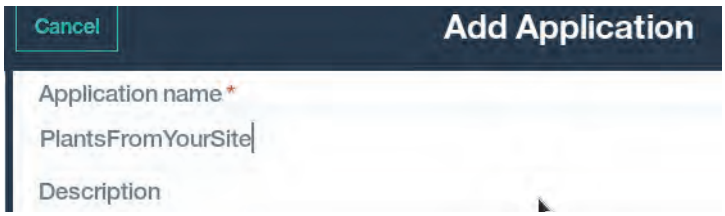
2. Click the plus (**+**) sign to create a new application.



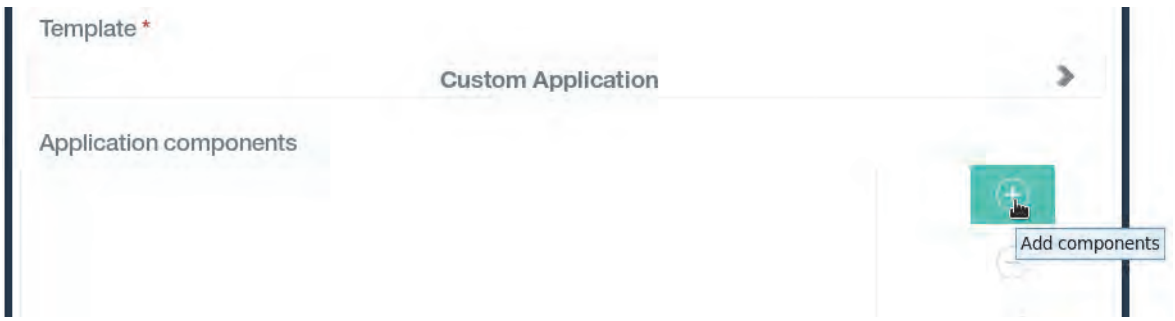
The Add Application window opens.



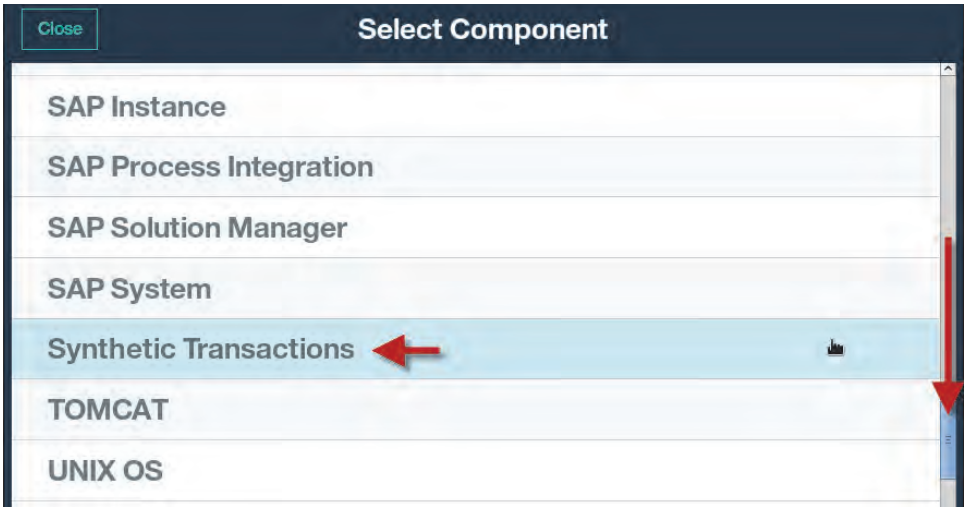
3. Enter **PlantsFromYourSite** in **Application Name**.



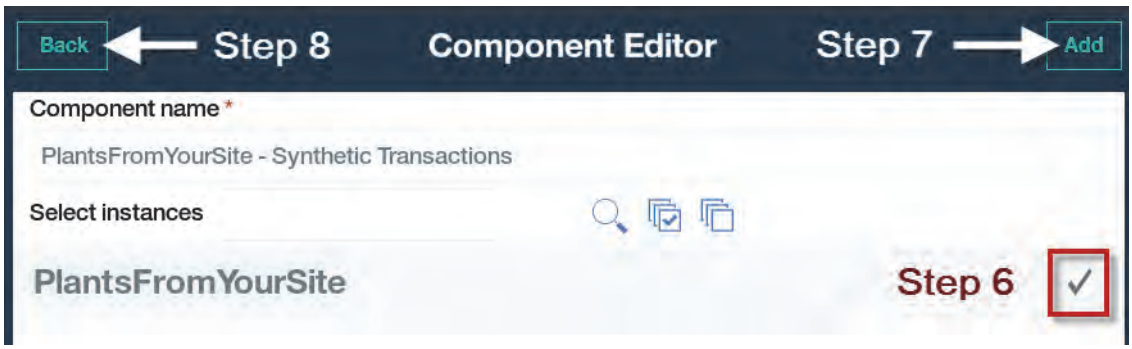
4. In the Application components dialog, click the plus (**+**) sign **to add a component**.



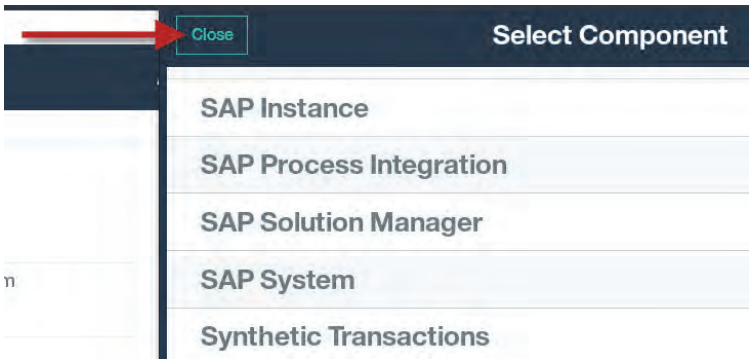
5. Scroll down to **Synthetic Transactions** in the **Select Component** dialog box and click it.



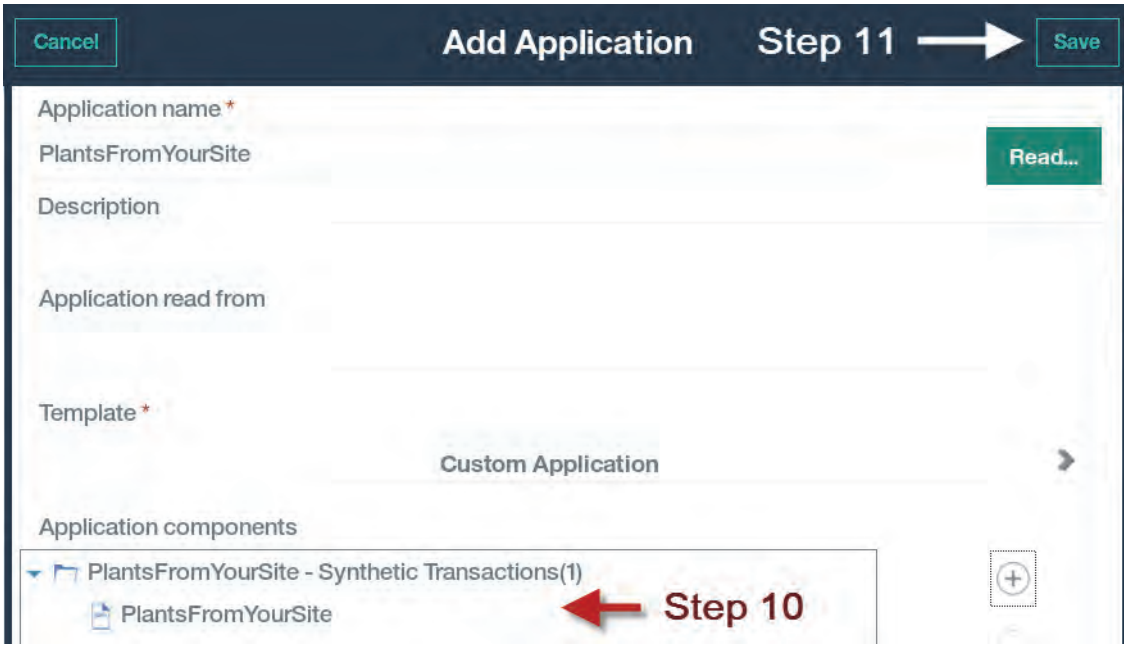
6. In the Component Editor window, select the **PlantsFromYourSite** synthetic transaction.



7. Click **Add** to associate the synthetic transaction with the application.
8. Click **Back** to return to the Select Component window.
9. Click **Close** to exit the Select Component window.

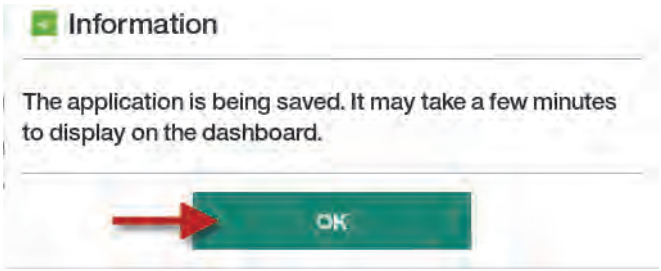


10. Confirm that the **PlantsFromYourSite** synthetic transaction was added.

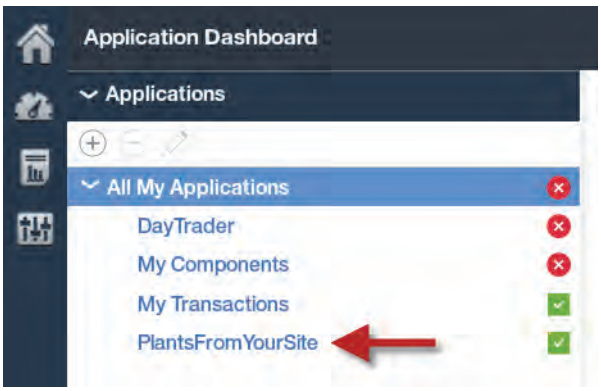


11. Click **Save**.

12. Click **OK** to complete the operation.

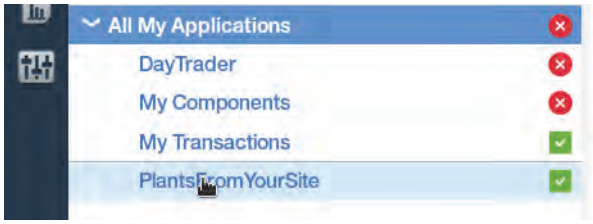


You are returned to the Application Dashboard and the **PlantsFromYourSite** application is displayed.

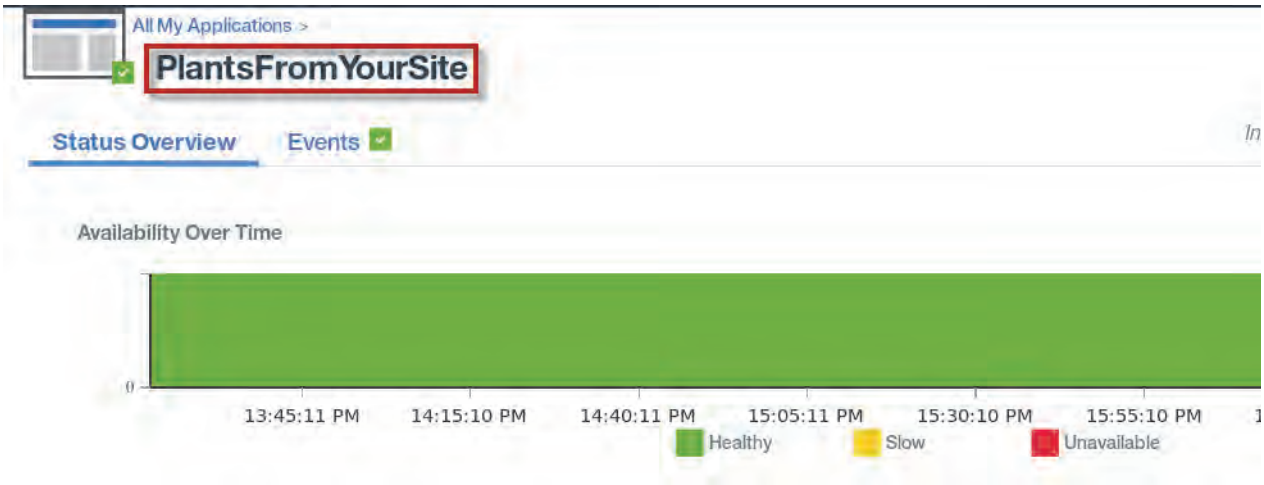


Exercise 3 Viewing synthetic transactions

1. Click the **PlantsFromYourSite** application in the **All My Applications** list.



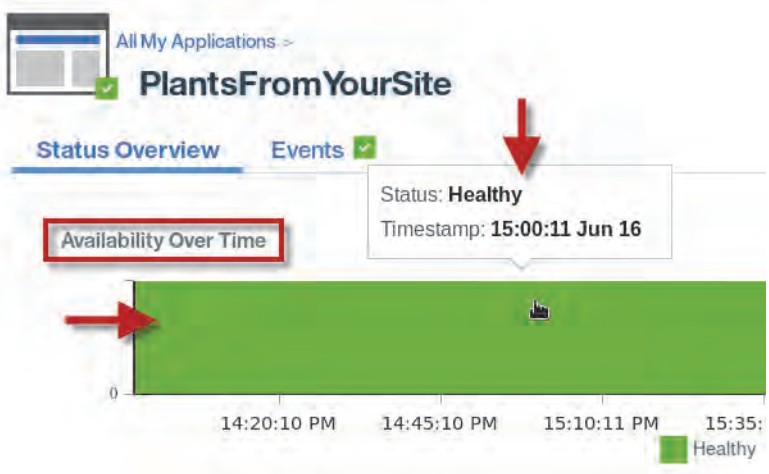
2. Synthetic transaction data is first displayed in the Availability Over Time widget.



This screen capture shows a few hours of data because of when this system was set up. Your system shows much less data.

In this case, *green* indicates that all transactions are completing in less time than their threshold value. *Yellow* indicates transactions that are taking longer than the threshold value. *Red* indicates that some transactions are failing.

3. Roll your cursor over the bar chart and see the average times for all transactions in that aggregation period.



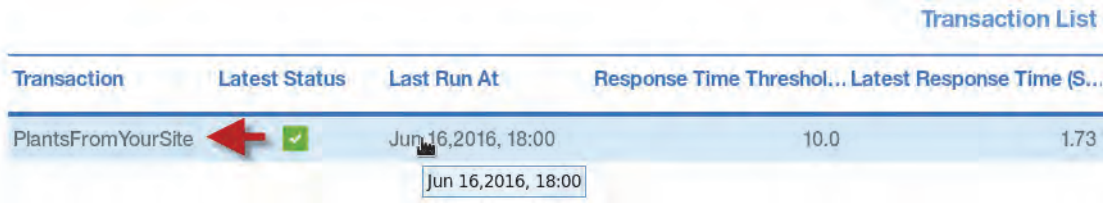
4. Click the **Availability Over Time** widget.

   The **Synthetic Transactions** page is displayed.

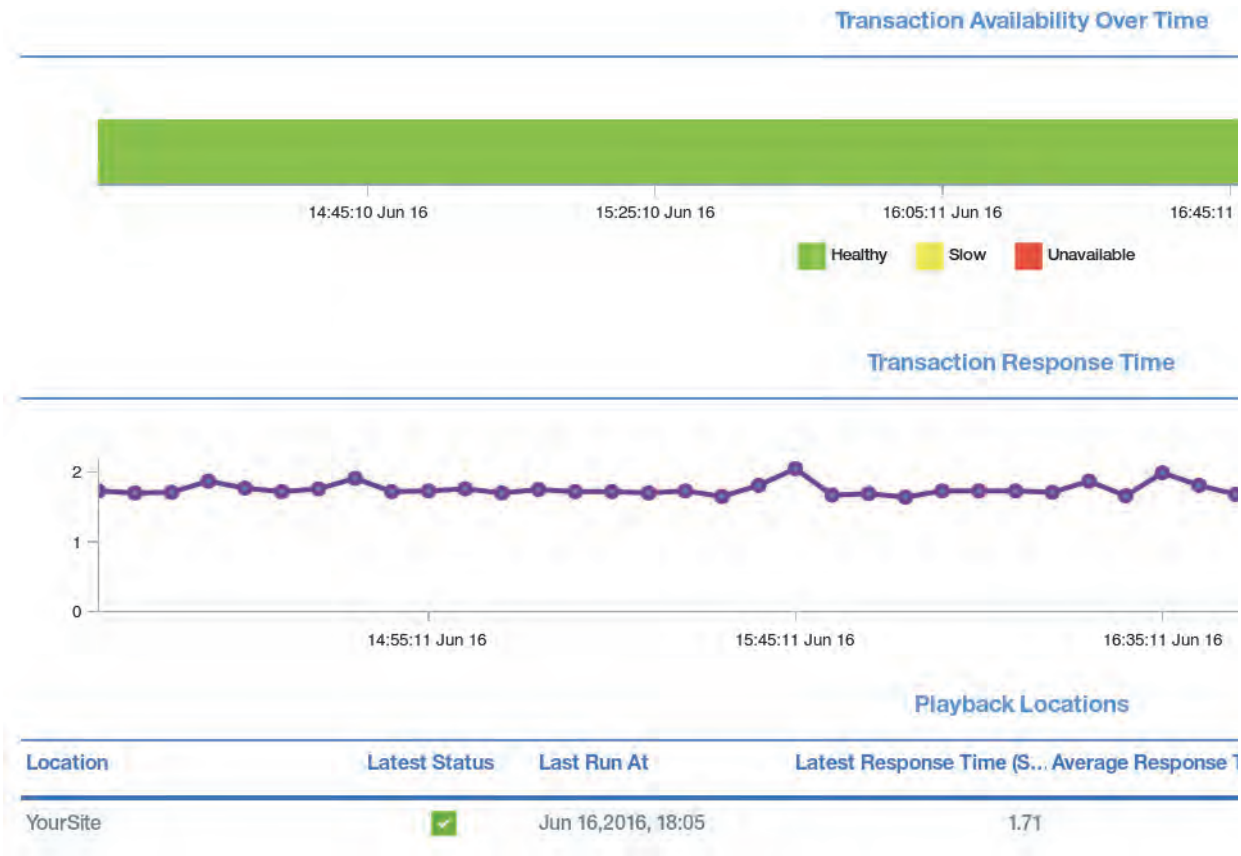
Examine the performance data for the transactions within Transactions List to determine whether any are having performance problems. When the same script is run from multiple locations. compare performance by location in the Location List. When multiple scripts are run from the same location, compare their performance in the Transaction List.

Examine the locations where scripts are playing from and their performance data to see whether any one location is having performance problems. If you deploy from multiple locations, you can see whether any specific location is causing the problems.

5. Click the **PlantsFromYourSite** transaction to open the Synthetic Transaction window for more detail.



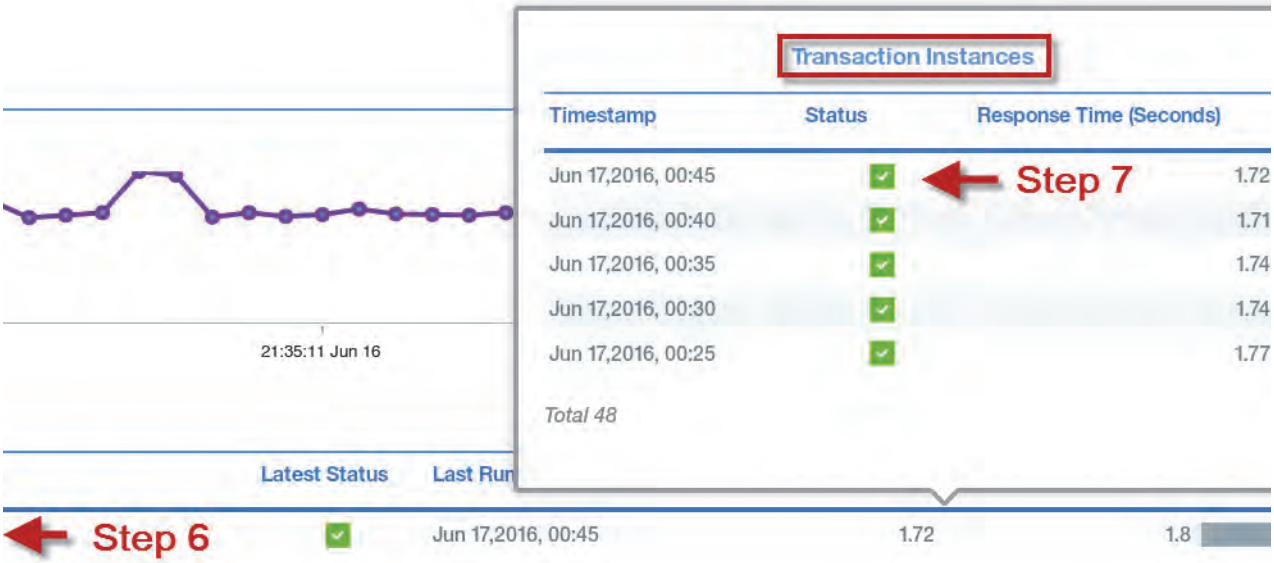
The Synthetic Transaction page opens.



This page shows performance and availability data of this specific script over time.

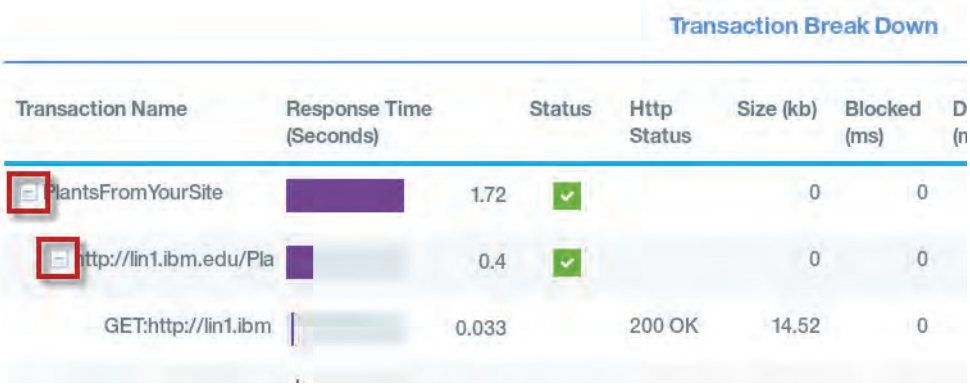
This sample transaction has no subtransactions. For transactions with subtransactions, more detail is available in the **Subtransactions** widget at the bottom of this dashboard.

6. Click the **YourSite** location to access a list of **Transaction Instances** for your location.

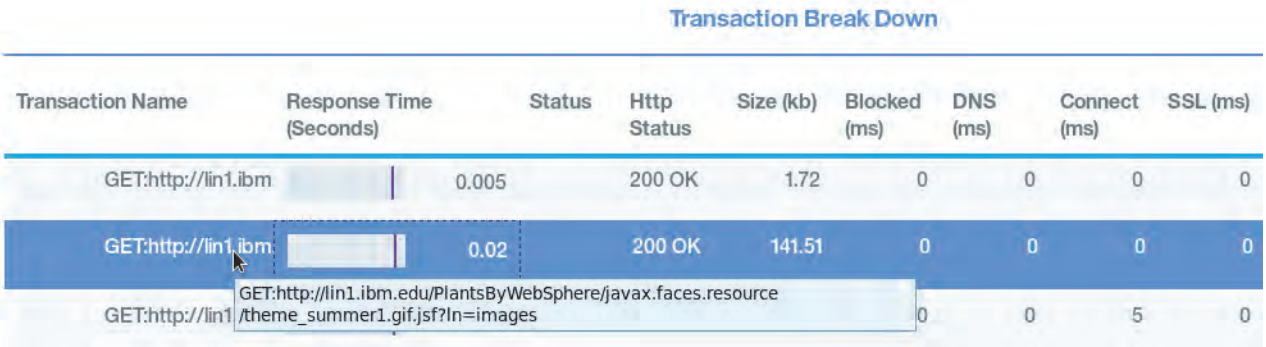


7. Click a **Transaction Instance** in the popup.

8. Click any plus (**+**) sign to expand the **Transaction Break Down**.



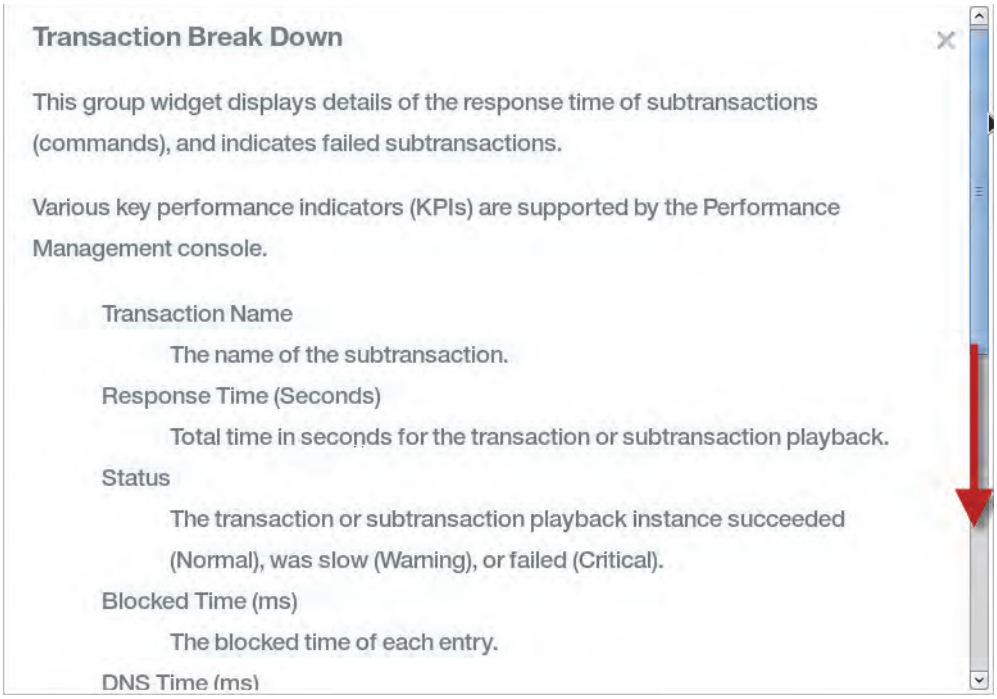
9. Scroll through the **Transaction Break Down** and click transaction events that have high response times or other indications of poor performance.



10. Hover over items in the **Transaction Name** column and review URL details in the flyovers.

11. Review the available metrics in the column header row.

12. Click the question mark (?) icon and scroll through the help window for more details of the available metrics.



6 Appendix A

In this lab session, you use IBM Application Performance Management to monitor Node.js. Node.js is a software platform that employs JavaScript for server-side solutions, these solutions are often used for receiving and responding to HTTP requests. The Node.js agent can be used to measure and collect data about the performance of Node.js applications. For example, throughput and response times for HTTP requests, and other measurements that relate to resource usage, are monitored and stored for display and analysis.

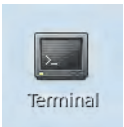
The Node.js agent is a single instance agent. It registers subnodes for each monitored Node.js application.

Objectives

After completing all the exercises, you can monitor Node.js application server resources.

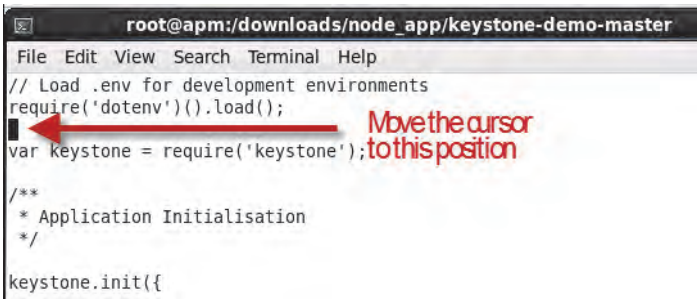
Exercise 1 Configuring the Node.js agent

1. On **lin2**, double-click the Terminal desktop icon to open a command prompt.



2. Navigate to the directory where the **keystone.js** file is located.
   ```
   cd /downloads/node_app/keystone-demo-master
   ```
3. Open the **keystone.js** file for editing.
   ```
   vi keystone.js
   ```

4. Use the arrow keys to move the cursor to the indicated position.



5. Type Shift-i to put the file into edit mode.

Three levels of monitoring are available with the Node.js agent:

- Resource data only

  `require('/opt/ibm/apm/agent/lx8266/nj/bin/plugin/knj_index.js');`
- Resource data plus diagnostic data

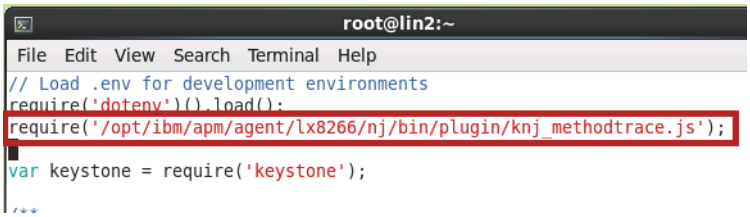
  `require('/opt/ibm/apm/agent/lx8266/nj/bin/plugin/knj_deepdive.js');`
- Resource data, plus diagnostic and method trace data

  `require('/opt/ibm/apm/agent/lx8266/nj/bin/plugin/knj_methodtrace.js');`

6. Type the following Require statement.
   `require('/opt/ibm/apm/agent/lx8266/nj/bin/plugin/knj_methodtrace.js');`

Attention: Be sure to use single quotation marks in the added REQUIRE statement.

7. Verify that your file is identical to the following screen capture.



8. Press Esc to exit the edit mode.
9. Type **:wq!** to exit **vi** and save the file.
10. Type the following command in the terminal window and press Enter.
    `/opt/ibm/apm/agent/bin/nodejs-agent.sh stop`
11. Wait for the command to complete before proceeding.
12. Type the following command in the terminal window and press Enter.
    `/opt/ibm/apm/agent/bin/nodejs-agent.sh start`
13. Wait for the command to complete before proceeding.

Exercise 2 Reviewing the conf.json file

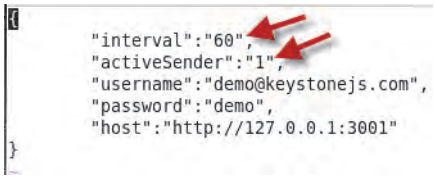
1. Type the following command to navigate to the directory containing the agent installation packages.

   `cd /downloads/node_app/httpSender/`

2. Type the following command to open **conf.json** for editing.

   `vi conf.json`

3. Review the first two parameters in the file.



   Currently, the file is configured to open 1 **activeSender** threads at an **interval** of 60 seconds.

4. Type the following command to exit the file without making any changes.

   `:q!`

5. Type the following command to open sendURLWithCookies.py for editing.

   `vi sendURLWithCookies.py`

6. Scroll down to the **self.getUrls** item. You can also type this command to search for the item:

   `/self.getUrls`
   `Press Enter`

```
class HttpRequestSender(threading.Thread):
  def __init__(self,activeSender=1,interval=1,host='http://127.0.0.1:3001'):
        threading.Thread.__init__(self)
        self.interval = interval
        self.host = host
        self.activeSender = activeSender
        self.postids = []
        self.username = 'demo@keystonejs.com'
        self.password = 'demo'
        self.getUrls = ['/keystone','/blog','/gallery','/contact','/keystone/sig
nin','/keystone/posts/','/keystone/galleries','/keystone/enquiries','/keystone/u
sers','/keystone/things']
        self.postUrl = '/keystone/posts'
```

   Currently, every time the Python script runs, it gets 11 URLs from the Keystone application, posts 10 times to the chat area, and deletes those 10 posts. The list of URLs might be edited further to reduce application load.

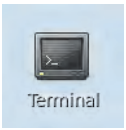
7. Type the following command to exit the file and make no changes.

   `:q!`

Exercise 3 Generating Node.js traffic

In this exercise, you start MongoDB, the Keystone.js application, and a script to generate Node.js traffic.

1. Double-click the Terminal desktop icon on the **lin2** VM to open a command prompt.



2. Type the command to start the Mongo database. Press Enter.

   `mongod -f /etc/mongod.conf`

3. Wait until you see a forked process number.

```
[root@apm ~]# mongod -f /etc/mongod.conf
about to fork child process, waiting until server is ready for connections.
forked process: 25867
child process started successfully, parent exiting
```

4. Navigate to the directory where the Node.js application is located.

   `cd /downloads/node_app/keystone-demo-master`

5. Type the command to start the Keystone application. Press Enter.

   `node keystone.js`

6. Ignore the **c++** error message and wait until the application initializes.

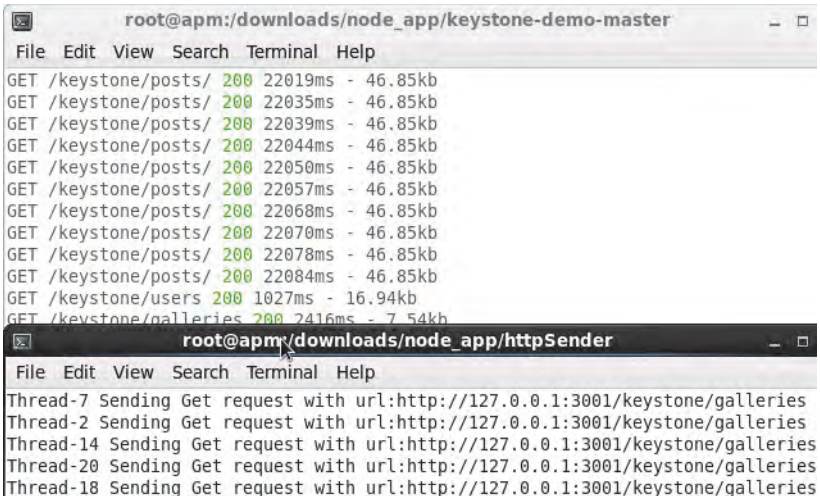
7. Open a separate terminal window, and type the command to navigate to the folder containing the traffic generation script. Press Enter.

   `cd /downloads/node_app/httpSender/`

8. Type the command to start the script. Press Enter.

   `./sendUrlWithCookies.py`

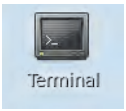
9. Review the flow of traffic.



Exercise 4 More configuration options

You can fine-tune several Node.js application monitoring parameters by editing the **plugin_*port*_conf.json** file. The port for the keystone.js application is 3001.

1. On lin2, double-click the Terminal desktop icon to open a command prompt.



2. Navigate to the directory where the **plugin_3001_conf.json** file is located.
   `cd /opt/ibm/apm/agent/lx8266/nj/bin/plugin/lib/`

3. Open the **plugin_3001_conf.json** file for editing.
   `vi plugin_3001_conf.json`

4. Type the following command and press Enter to move to the end of the file.
   `:$`

5. Review the current settings in the file with the data collector settings in the table.

```
    "traceFile": "/tmp/app.log",
    "traceLevel": "error",
    "traceSizeRotate": "10",
    "deepDive": {
        "enabled" : true,
        "minClockTrace": 1,
        "minClockStack": 100,
        "eventsPerFile": 200,
        "fileCommitTime": 60,
        "maxFiles": 20,
        "sampling":10
    }
}
```

Table 1 Data collector settings

Diagnostic data category	Property	Action
Enable/Disable collection of diagnostics data	enabled	Set to true to enable, otherwise set to false, the default value is true.
Minimum time delta for stack trace reporting	minClockStack	Set to a value in milliseconds
Minimum time delta to report requests	minClockTrace	Set to a value in milliseconds
Maximum number of events per file	eventsPerFile	Set to a maximum number of events value
Maximum amount of time to report to one file	fileCommitTime	Set to the maximum time in seconds

Table 1 Data collector settings

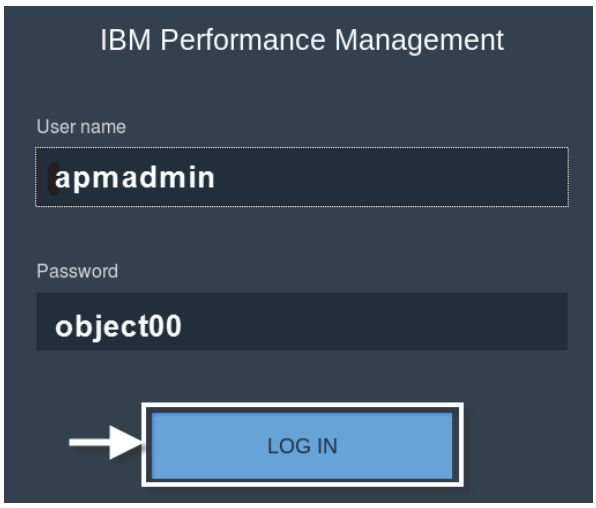
Diagnostic data category	Property	Action
Maximum number of files to keep before the oldest ones are deleted.	maxFiles	Set to the maximum number of files
Request sampling period	sampling	Set to the wanted sampling period. The default value is 10. A value of 10 means that the agent collects one of every 10 requests.

Advanced configuration options are not used for this course.

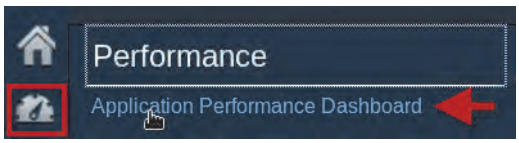
6. Type the following command to exit the file and make no changes.
   `:q!`

Exercise 5 Logging in to the Performance Management console

1. Move to the **Performance Management console** on the **apm** image.
2. If necessary, log in as **root** user with password **object00**.

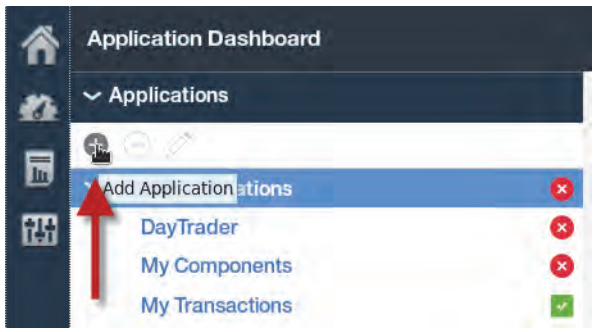


3. Navigate to the Application Performance Dashboard by clicking the highlighted icon.

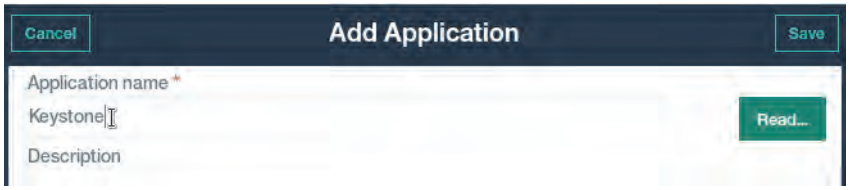


Exercise 6 Creating the Keystone application in the Performance Management console

1. Click the plus (**+**) sign in the **Applications** widget.

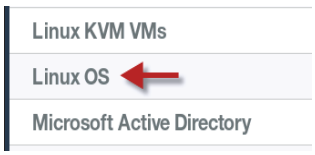


2. Type **Keystone** in the **Application name** field.

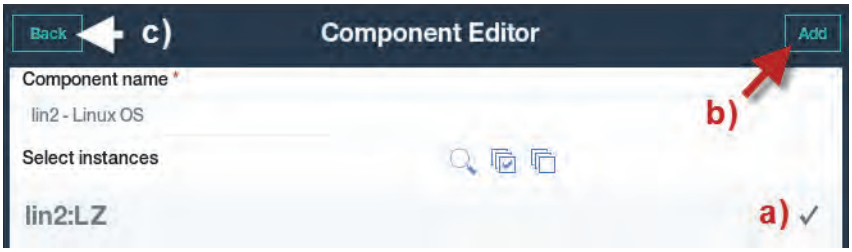


3. Click the plus (**+**) sign to add a component.

4. Select **Linux OS** in the list of available components.



5. Add the single instance to the application.



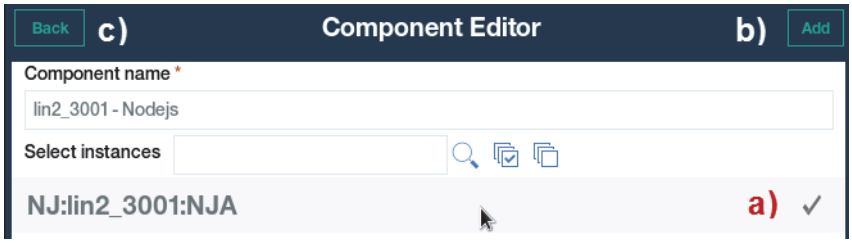
   a. Click the instance.

   b. Click **Add**.

c. Click **Back** to complete the operation.

6. Select **Node.js** in the list of available components.

7. Add the single instance to the application.

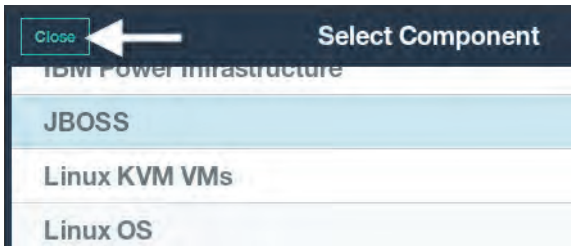


a. Click the instance.

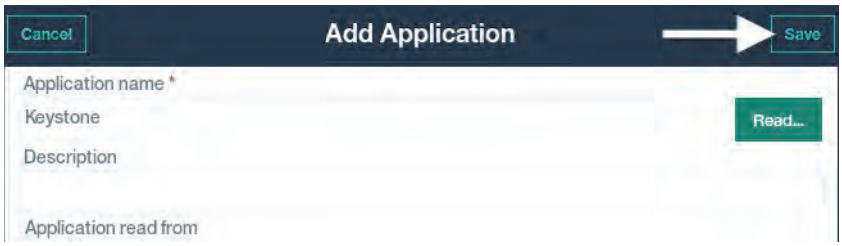
b. Click **Add**.

c. Click **Back** to complete the operation.

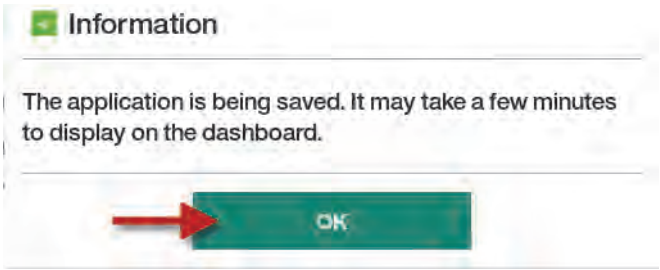
8. Click **Close**.



9. Click **Save**.

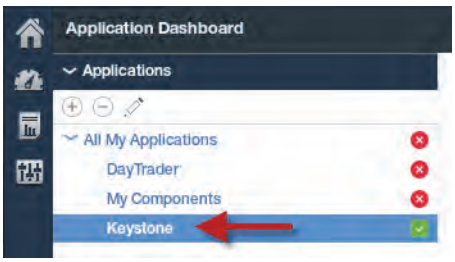


10. Click **OK** to complete the operation.

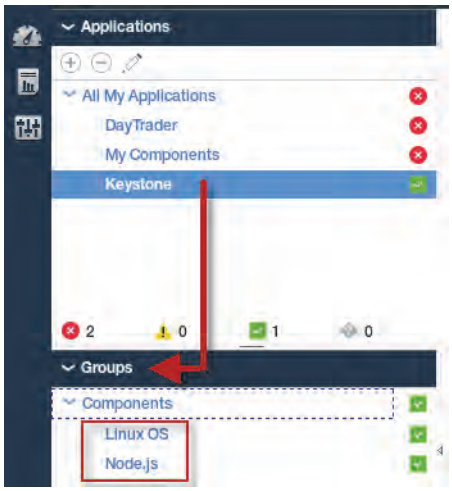


You are returned to the Application Dashboard and the **Keystone** application is displayed.

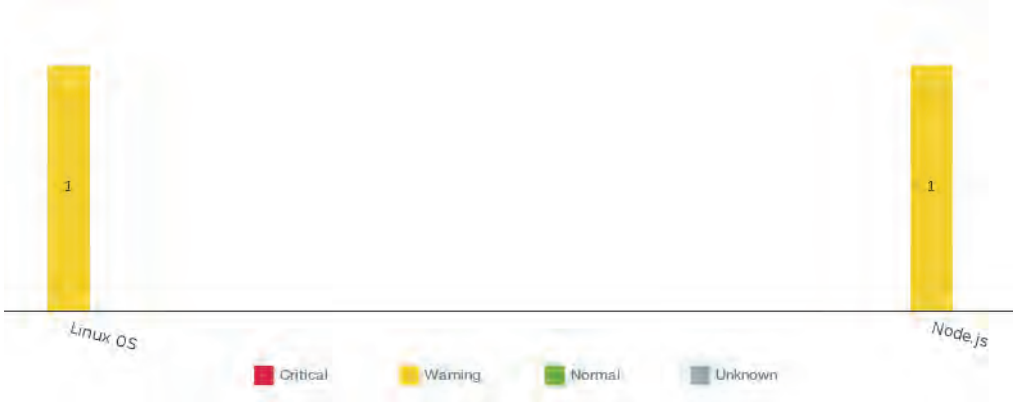
11. Confirm the creation of your application.



12. Verify that your Node.js agent is listed in the **Groups** widget.



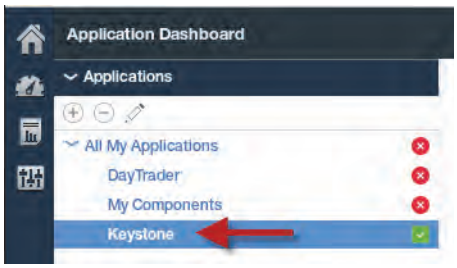
13. Verify that your components are listed in the **Current Components Status** widget.



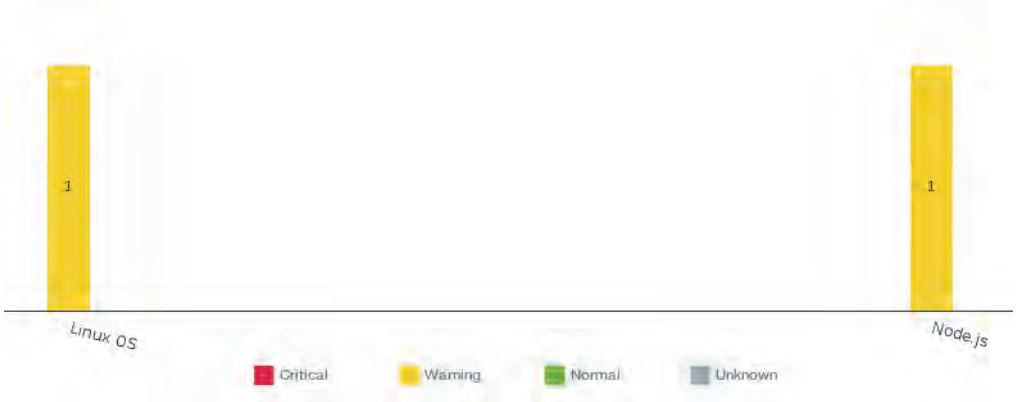
Note: The status of your agents might be different.

Exercise 7 Monitoring Node.js resources

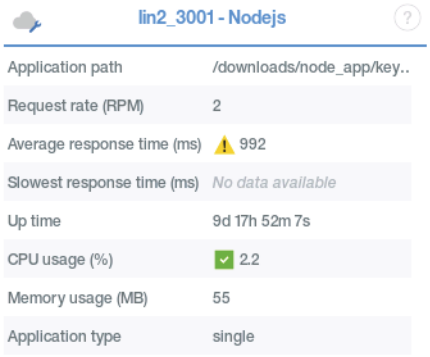
1. Click the **Keystone** application to return to the **Status Overview** dashboard.



2. Click the **Node.js** bar in the **Current Components Status** widget.

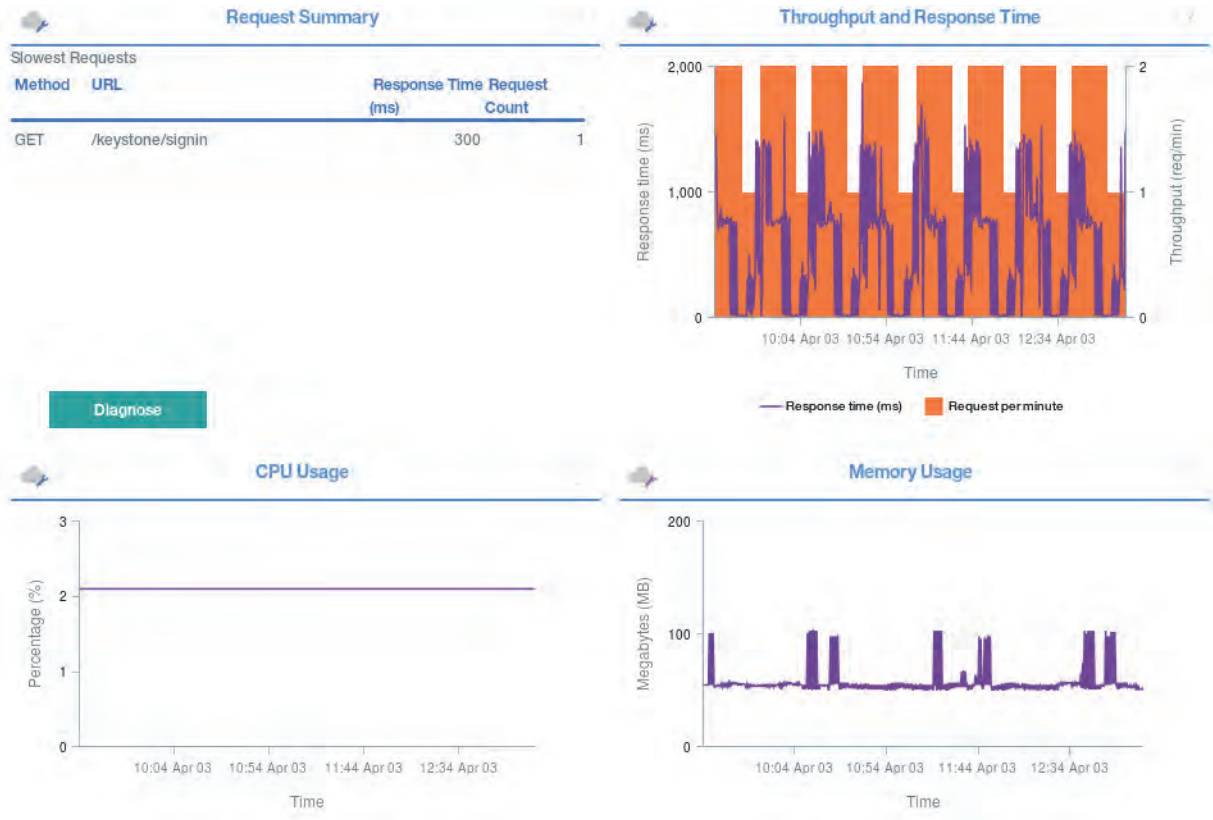


3. Click the Node.js instance widget.



4. Confirm the presence of activity in the **Request Summary** and **Throughput and Response Time** widgets.

5. Review the widgets on the **Status Overview** dashboard.



Currently, these widgets are supported:

- Request Summary

  Returns Method type, request count, response time, and URL

- Throughput and Response Time

  Returns requests per minute, response time, and time

- CPU Usage

CPU usage history of the Node.js application

- Memory Usage

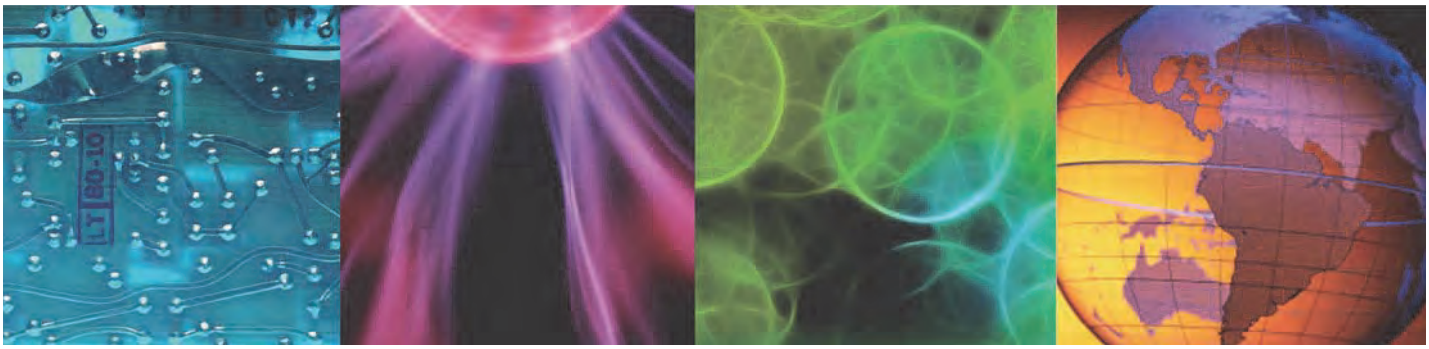
  Memory usage history of the Node.js application

- Status Group

  Use to view a comma-separated list of KPI names whose values contribute to a warning or critical status for the managed resource.

Note: No additional dashboards for Node.js resource widgets are available currently.

TOD45 ERC 1.0



ibm.com/training

Authorized IBM Training