



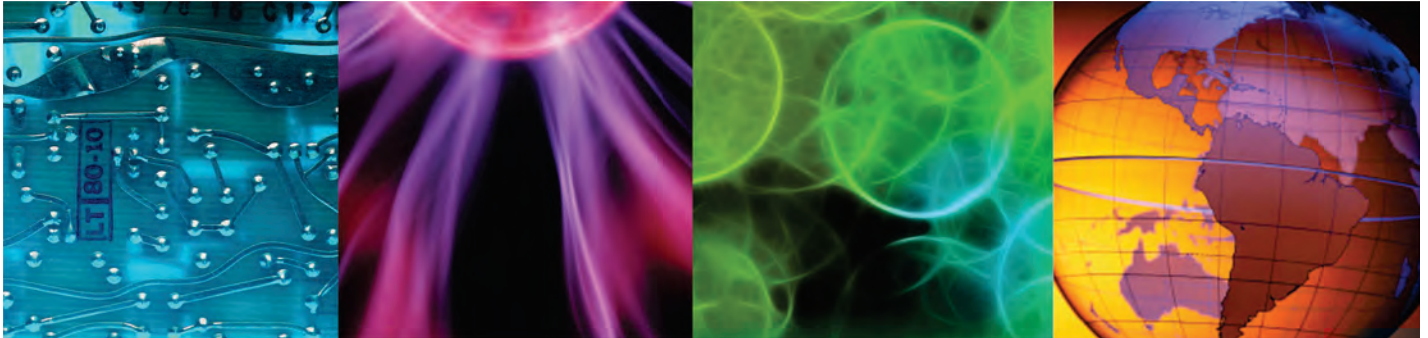
# IBM Training

## **IBM Application Performance Management Advanced 8.1.3 Fundamentals**

### **Course Exercises**

Course code TOD45 ERC 1.0

August 2016



All files and material for this course are IBM copyright property covered by the following copyright notice.

© Copyright IBM Corp. 2016. All Rights Reserved.

US Government Users Restricted Rights: Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.



# Contents

<b>About these exercises</b> .....	<b>iv</b>
<b>1 Monitoring with IBM Application Performance Management Advanced</b> .....	<b>1</b>
Objectives .....	1
Exercise 1 Starting DB2 .....	2
Exercise 2 Modifying the gdc_custom.properties file .....	2
Exercise 3 Starting the DayTrader script .....	6
Exercise 4 Accessing the Performance Management console .....	9
<b>2 Application resource monitoring</b> .....	<b>11</b>
Objectives .....	11
Exercise 1 Monitoring WebSphere resources .....	11
Exercise 2 Monitoring the heap .....	23
<b>3 Code-level monitoring</b> .....	<b>27</b>
Objectives .....	27
Exercise 1 Accessing code-level data for WebSphere .....	28
Exercise 2 Generating more traffic .....	31
<b>4 Transaction tracking</b> .....	<b>37</b>
Objectives .....	37
Exercise 1 Exploring aggregate topologies .....	37
Exercise 2 Exploring transaction instance topologies .....	41
<b>5 Synthetic transaction and user monitoring</b> .....	<b>49</b>
Objectives .....	49
Exercise 1 Configuring a synthetic transaction .....	49
Exercise 2 Creating a synthetic application .....	51
Exercise 3 Viewing synthetic transactions .....	55
<b>6 Appendix A</b> .....	<b>61</b>
Objectives .....	61
Exercise 1 Configuring the Node.js agent .....	61
Exercise 2 Reviewing the conf.json file .....	63
Exercise 3 Generating Node.js traffic .....	64
Exercise 4 More configuration options .....	66
Exercise 5 Logging in to the Performance Management console .....	67
Exercise 6 Creating the Keystone application in the Performance Management console .....	68
Exercise 7 Monitoring Node.js resources .....	71



# 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

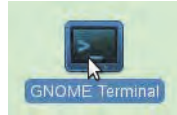
After completing this unit, you can perform the following tasks:

- 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
```

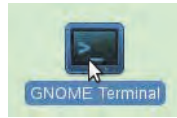
- To enable method data collection at server startup, set the property to true.

```
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
```

```
lin1:~ # 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.

```
#####  
#  
G  
/dc.sampling.enable
```

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.

```
#####  
#dc.sampling.enable=true  
#####  
#  
# The following property deter  
# You can make it smaller  
# Tinkering with this is no  
all  
# in size  
#####  
#dc.sampling.mapsize=5000  
-- INSERT --
```

6. Delete the **#** character.
7. Press **End** to move to the end of the line and set the parameter to **false**.

```
#####  
dc.sampling.enable=false  
#####
```

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

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

```
#####  
#  
# The following property determines the sampling map size  
# You can make it smaller to reduce memory foot print.  
# Tinkering with this is not recommended as the objects stored are very small  
# in size  
#####  
#dc.sampling.mapsize=5000  
/dc.sampling.methsampler.enabled
```

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

```
#####  
#  
# The following property indicates if DFE mode needs to enable  
# sampling for method tracing. This property is effective only when  
# dfe.enable.methoddata=true; When dfe.enable.methoddata and dc.sampling.methsampler.enabled  
# are true, method data is captured automatically for some requests on  
# certain violations, errors or capture window  
# Note: This determines for a sampled request if method trace needs to be  
# collected or not.  
#####  
#dc.sampling.methsampler.enabled=true  
#####
```

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.

```
#####  
dc.sampling.methsampler.enabled=false  
#####
```

15. Press **Esc** to exit **Insert** mode.  
16. Type **/dfe.enable.methoddata** and press **Enter** to find the parameter.

```
#####  
#  
# The following property:  
/dfe.enable.methoddata
```

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

```
#####  
#  
# The following property indicates if DFE mode needs to enable application  
# Method tracing. This property can also be enabled after DC is started  
#####  
#dfe.enable.methoddata=true  
#####
```

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**.

```
#####  
ife.enable.methoddata=true  
#####
```

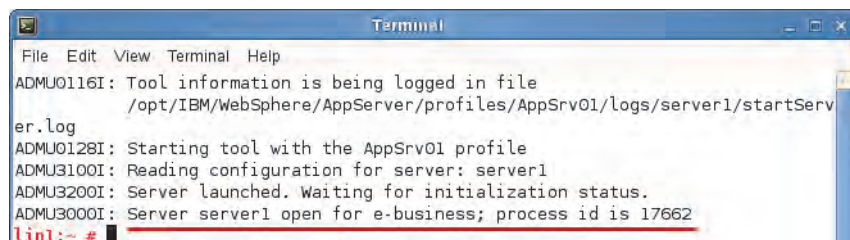
21. Press **Esc** to exit **Insert** mode.

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.

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", and "Help". The terminal output shows the following messages:  
ADMU0116I: Tool information is being logged in file  
          /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/startServ  
er.log  
ADMU0128I: Starting tool with the AppSrv01 profile  
ADMU3100I: Reading configuration for server: server1  
ADMU3200I: Server launched. Waiting for initialization status.  
ADMU3000I: Server server1 open for e-business; process id is 17662  
The prompt "lin1:~ # " is visible at the bottom of 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

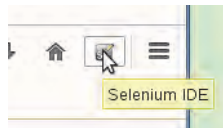
With IBM HTTP Server Response Time plug-in, top 10 resource timings are collected by default if your browser has JavaScript enabled. The Selenium IDE plug-in was installed in the Firefox browser on **lin1.ibm.edu** to capture traffic that the Response Time Agent can detect.

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

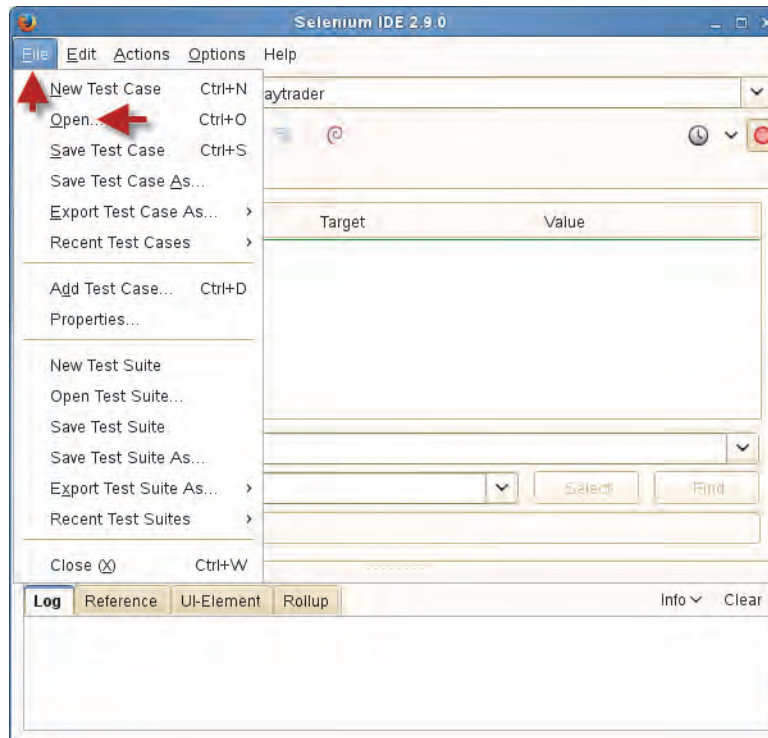
1. Open a Firefox Browser on **LIN1** by double-clicking the Firefox icon on the desktop.



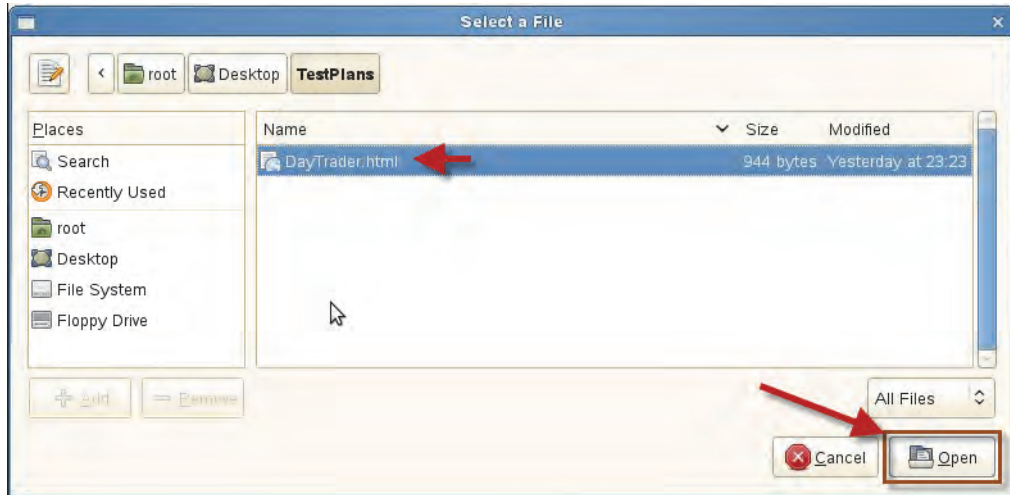
2. Click the Selenium IDE icon in the upper right of the browser window.



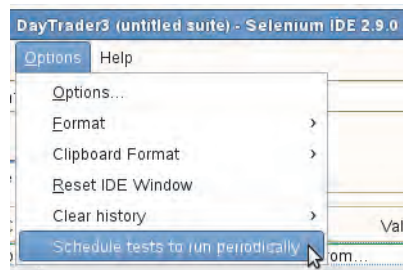
3. Click **File > Open**.



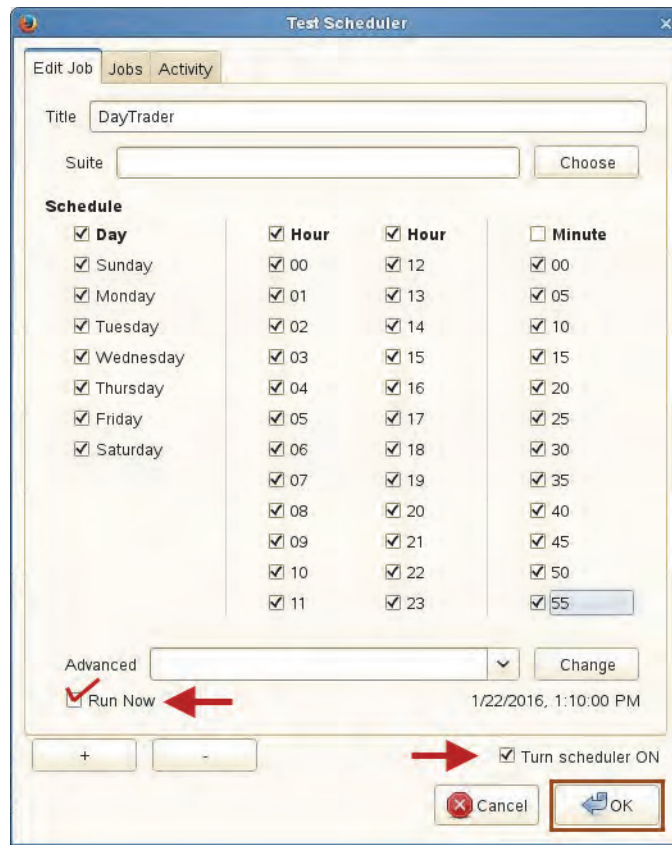
4. Select the **DayTrader** file in the **TestPlans** folder on the desktop and click **Open**.



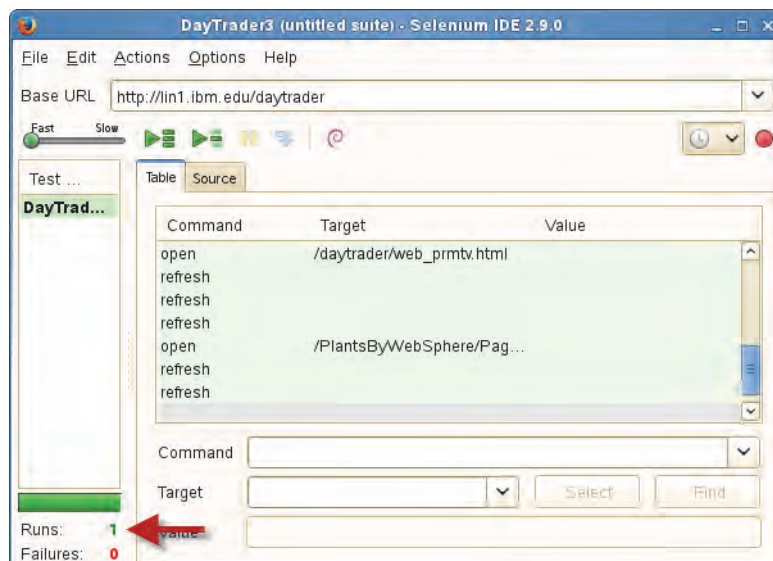
5. Select **Options > Schedule tests to run periodically** from the toolbar menu.



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



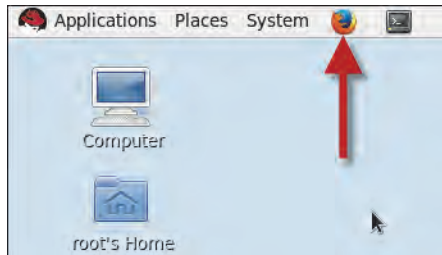
7. Confirm success of the script.



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**.



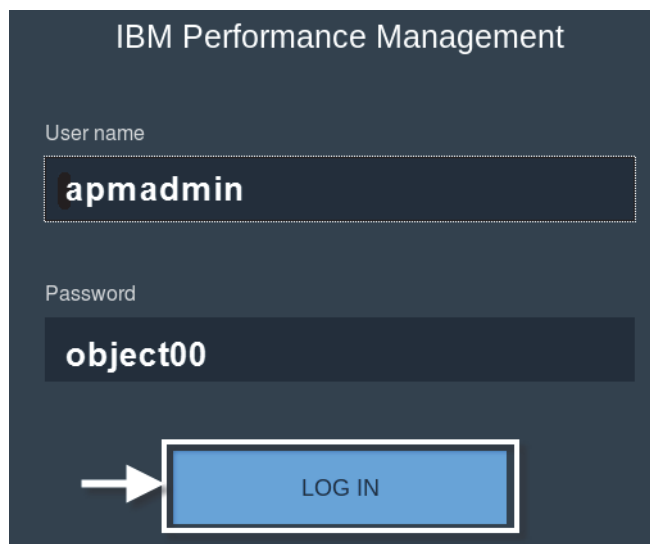
4. Click **Add Exception**.



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

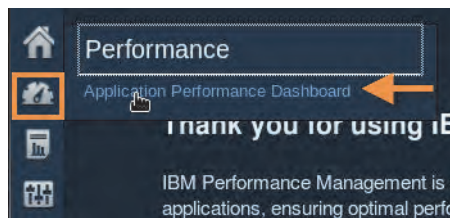
In this lab session, you use IBM Application Performance Management to monitor WebSphere resources.

### Objectives

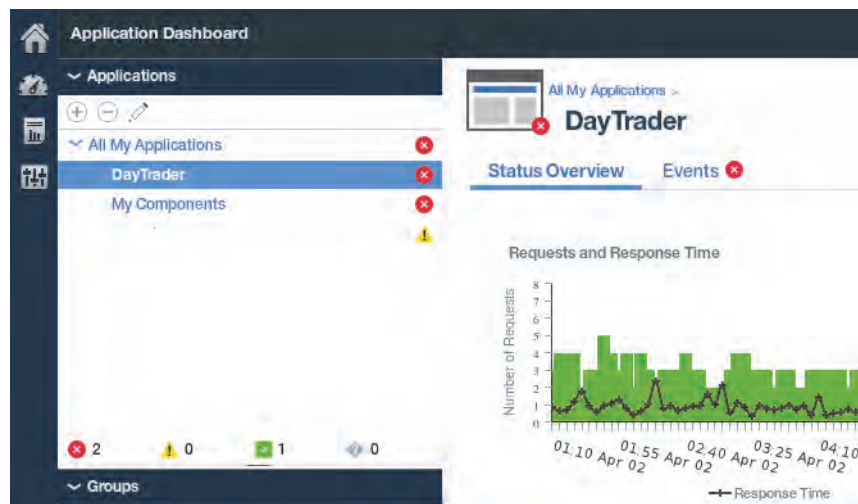
After completing all the exercises, you can monitor WebSphere Application Server resources by using the Performance Management Console

### Exercise 1 Monitoring WebSphere resources

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



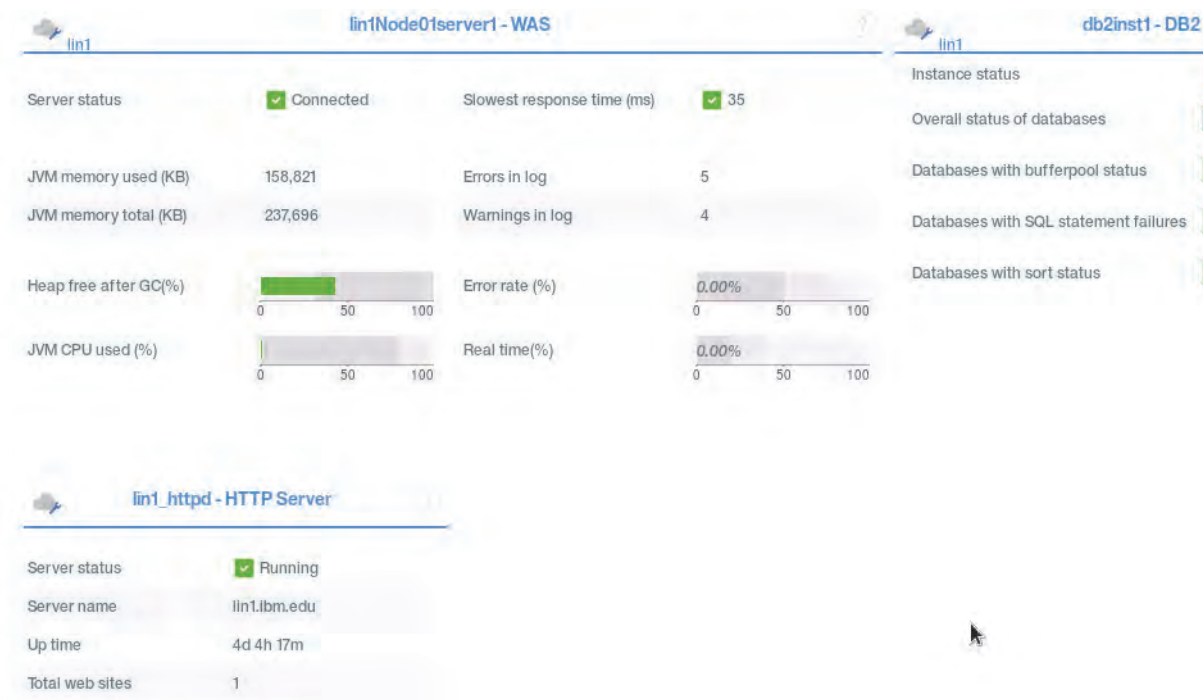
2. In the **Performance Management** console, select **DayTrader** in the application list.



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

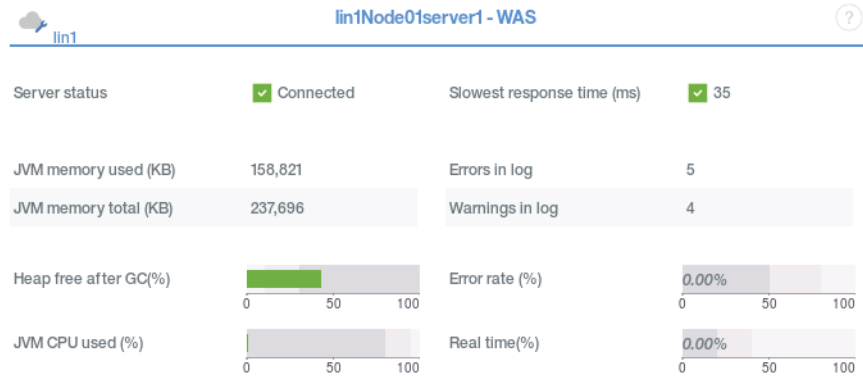


4. Review the component widgets on the **Status Overview** dashboard.





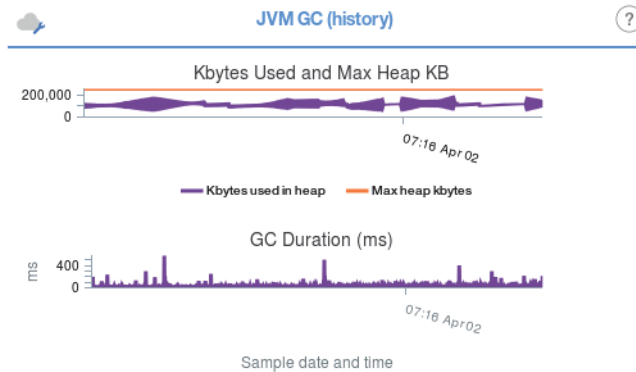
5. Click the **WebSphere** widget to access the resource dashboard.



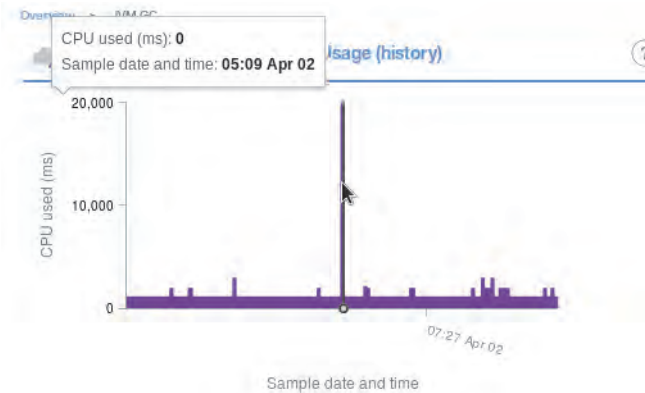
6. Review the resource widgets of the **WebSphere Status Overview** dashboard.



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



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



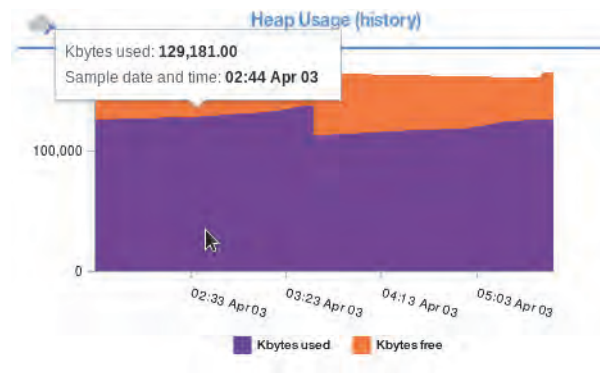
- Perform the same action on the **JVM Garbage Collection** and **Throughput and Average Response Time** widgets.



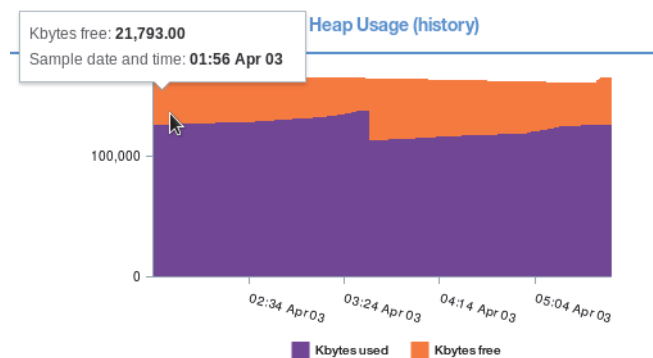
**Note:** On a lightly loaded lab system, none of the values in these charts are alarming. However, in a production environment, the ability to identify the precise time of a spike in CPU usage or Response Time is valuable.

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.

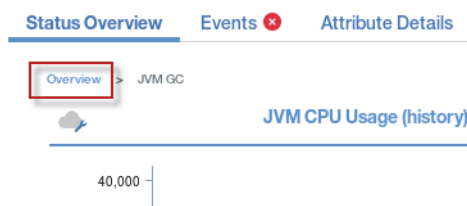
11. Move the mouse pointer over the purple data display area of the **Heap Usage (history)** chart to see values for the relevant attribute.



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

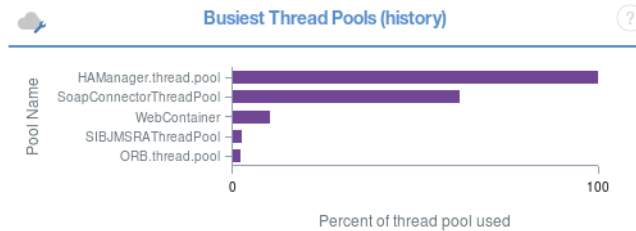


13. Just below the **Status Overview** tab, click the **Overview** link to return to the main WebSphere resource dashboard.



**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.

A table titled "Thread Pools" with columns: Thread Pool Name, Maximum Pool Size, Average Active Threads, and Average Pool Size. The Average Pool Size column is highlighted, indicating it is sorted. A red arrow points to the "Average Pool Size" header. The table data is as follows:

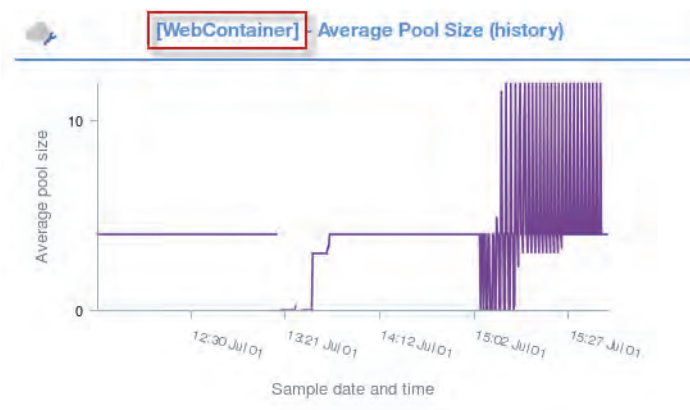
Thread Pool Name	Maximum Pool Size	Average Active Threads	Average Pool Size
WebContainer	50	1.0	6.999
SoapConnectorThreadPool	5	0.009	3.0
HAManager.thread.pool	2	0.0	2.0
ORB.thread.pool	50	0.0	1.0
SIBJMSRAThreadPool	41	0.0	1.0
AriesThreadPool	5	0.0	0.0
Default	20	0.0	0.0

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

A table titled "Thread Pools" with columns: Thread Pool Name, Maximum Pool Size, Average Active Threads, and Average Pool Size. The "WebContainer" row is highlighted in blue, indicating it is selected. The table data is as follows:

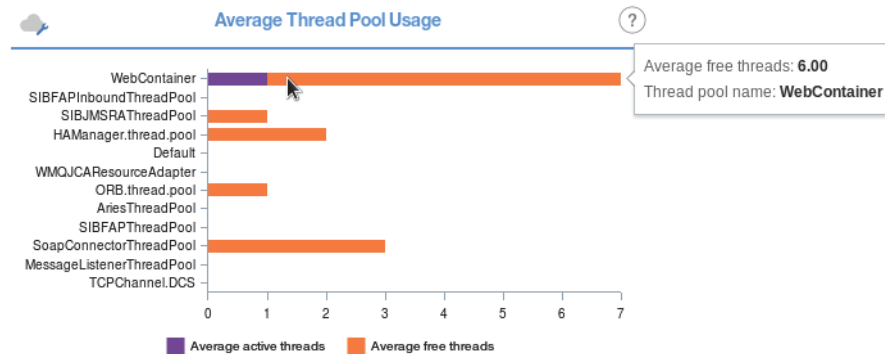
Thread Pool Name	Maximum Pool Size	Average Active Threads	Average Pool Size
WebContainer	50	1.0	6.999
SoapConnectorThreadPool	5	0.009	3.0
HAManager.thread.pool	2	0.0	2.0
ORB.thread.pool	50	0.0	1.0

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

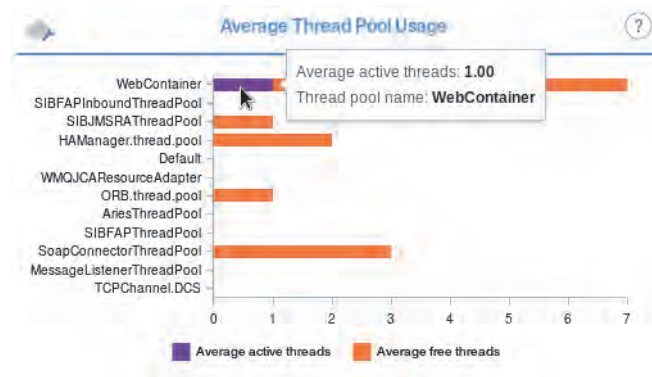


**Note:** Not all dashboard headers are given the prefix, including **Average Thread Pool Usage**.

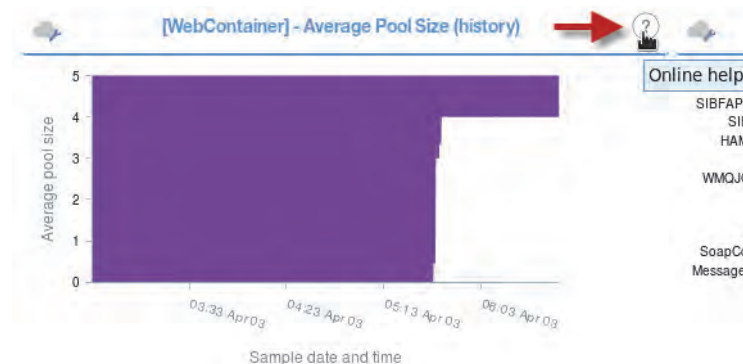
17. Move the mouse pointer over the orange section of the **WebContainer** bar in the **Average Thread Pool Usage** bar chart. Examine the flyover to see the **average free threads** for the selected pool for the current interval.



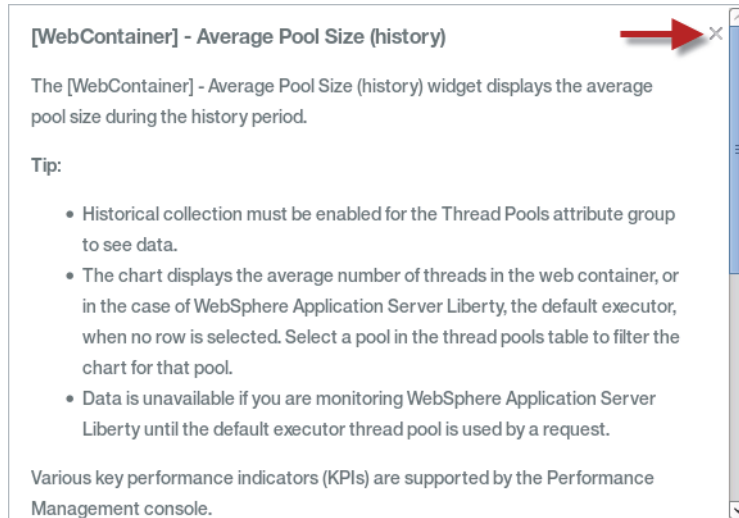
18. Move the mouse pointer over the purple section of the **WebContainer** bar in the **Average Thread Pool Usage** bar chart. Examine the flyover to see the **average active threads** for the selected pool for the current interval.



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

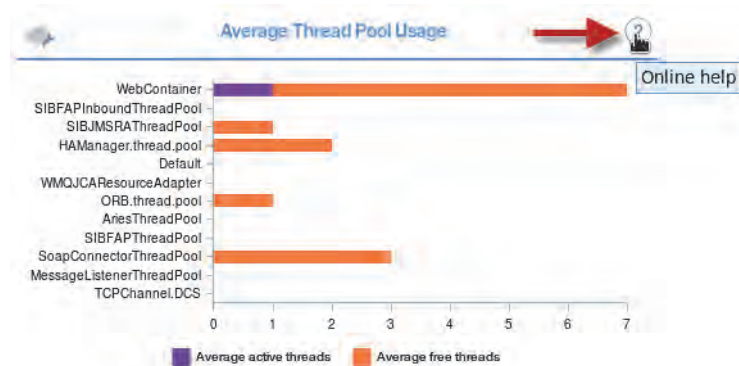


20. Review the help for the **Average Pool Size** widget.

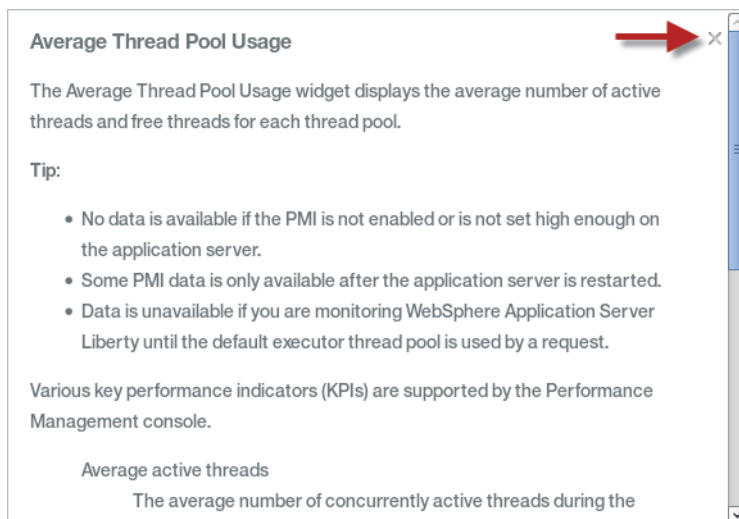


21. Click the **X** to close the window.

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

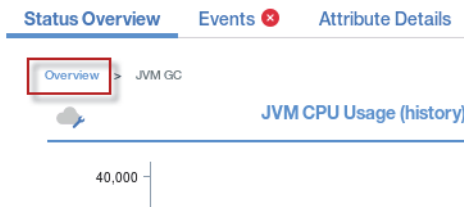


23. Review the help for the **Average Thread Pool Usage** widget.

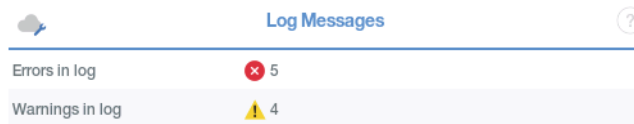


24. Click the **X** to close the window.

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

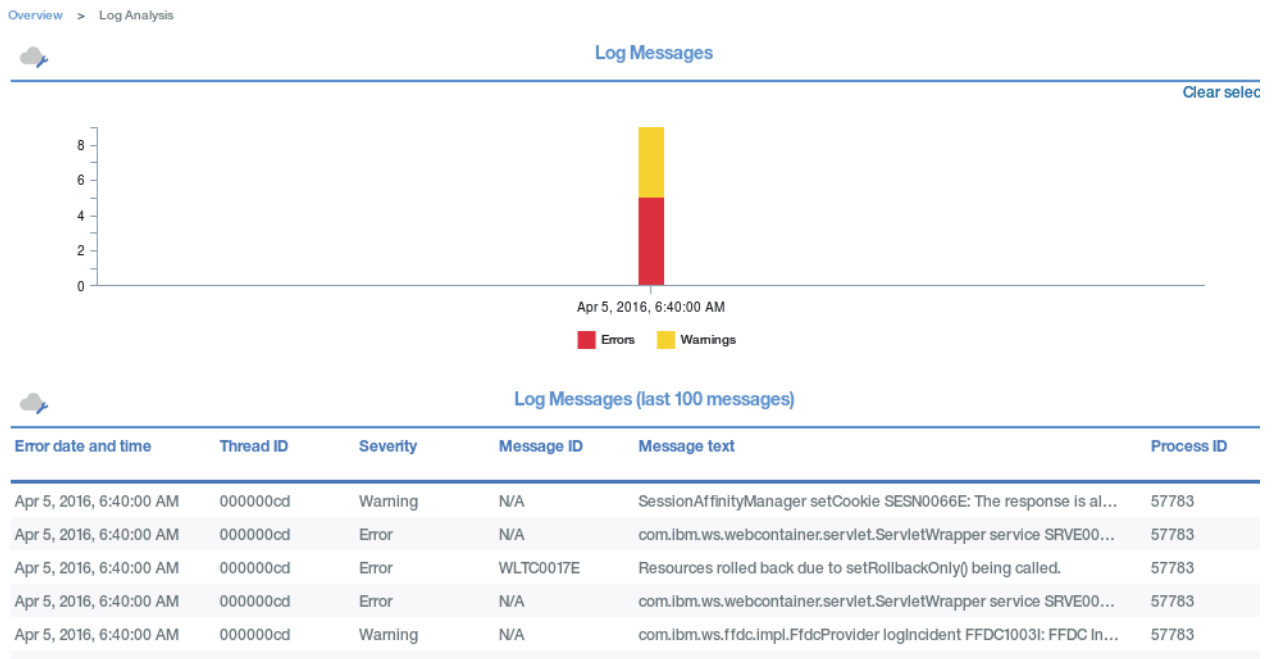


26. Scroll down to the **Log Messages** widget and click it.



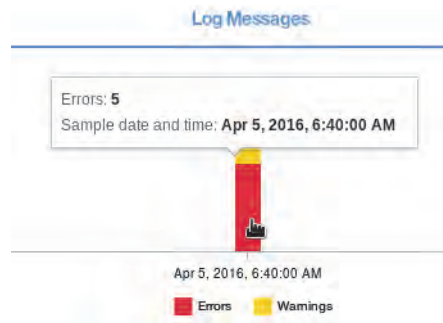
**Note:** On a lightly loaded system like your lab image, you might not see any messages. This behavior is normal. If you see no messages, proceed to [Step 32](#) on page 21.

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

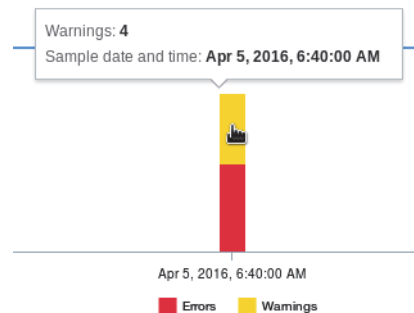




28. Hover over the red portion of the bar, if present, to see the number of **Errors** in the last 100 log messages.



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.

Log Messages - 2016-04-05T06:40:00Z

Error date and time	Thread ID	Severity	Message ID	Message text
Apr 5, 2016, 6:40:00 AM	000000cd	Warning	N/A	SessionAffinityManager se
Apr 5, 2016, 6:40:00 AM	000000cd	Warning	N/A	com.ibm.ws.ffdc.impl.Ffdd
Apr 5, 2016, 6:40:00 AM	000000cd	Warning	N/A	com.ibm.ws.webcontainer,

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

Log Messages - 2016-04-05T06:40:00Z

Message ID	Message text	Process ID
N/A	SessionAffinityManager setCookie SESN0066E: The response is al...	57783
N/A	com.ibm.ws.ffdc	
N/A	com.ibm.ws.web	
N/A	com.ibm.ws.webcontainer.srt.SRTServletResponse addHeader SR...	57783

SessionAffinityManager setCookie SESN0066E: The response is already committed to the client. The session cookie cannot be set.



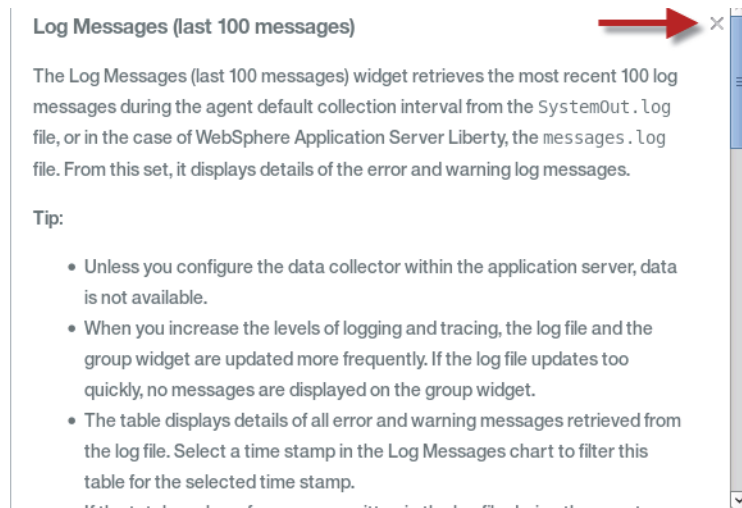
32. Click the question mark icon at the upper right of the **Log Messages** widget to view additional information.



Log Messages - 2016-04-05T06:40:00Z

Message ID	Message text	Process ID	Online help
N/A	SessionAffinityManager setCookie SESN0066E: The response is al...	57783	
N/A	com.ibm.ws.ffdc.impl.FfdcProvider logIncident FFDC1003I: FFDC In...	57783	

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



**Log Messages (last 100 messages)**

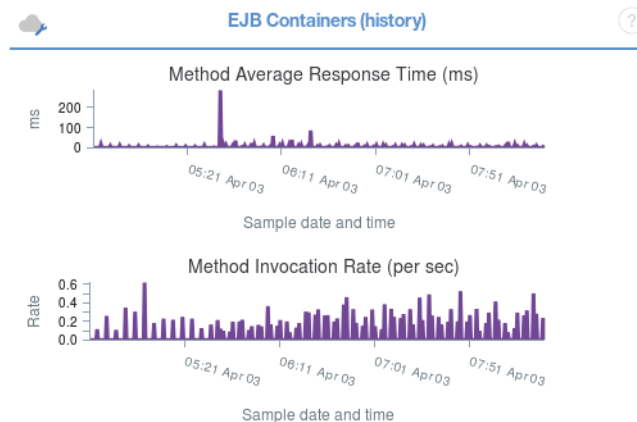
The Log Messages (last 100 messages) widget retrieves the most recent 100 log messages during the agent default collection interval from the SystemOut . log file, or in the case of WebSphere Application Server Liberty, the messages . Log file. From this set, it displays details of the error and warning log messages.

**Tip:**

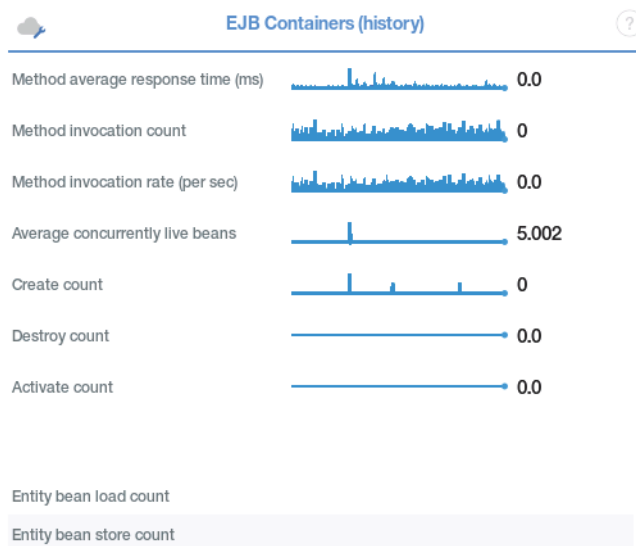
- Unless you configure the data collector within the application server, data is not available.
- When you increase the levels of logging and tracing, the log file and the group widget are updated more frequently. If the log file updates too quickly, no messages are displayed on the group widget.
- The table displays details of all error and warning messages retrieved from the log file. Select a time stamp in the Log Messages chart to filter this table for the selected time stamp.

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

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

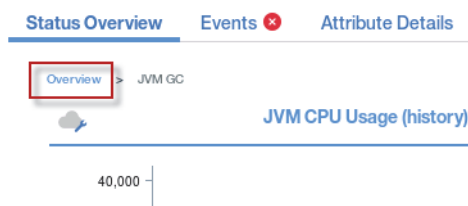


36. Pay particular attention to a widget with the same name, **EJB Containers (history)**, and note the greater level of detail provided.



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.

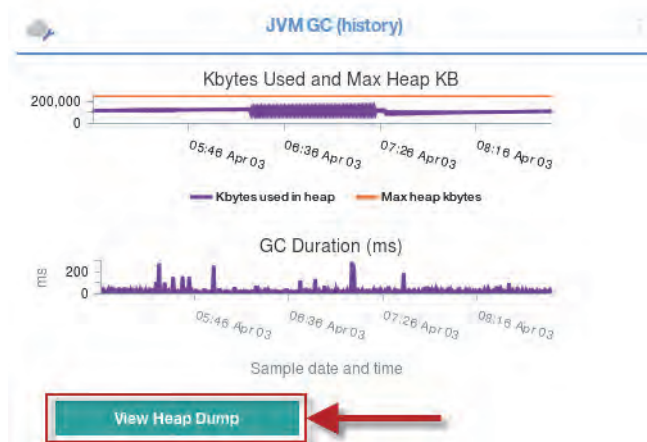


**Note:** No data might be available for some widgets, such as **Slowest Web Services**. On test systems such as these, that behavior is normal.

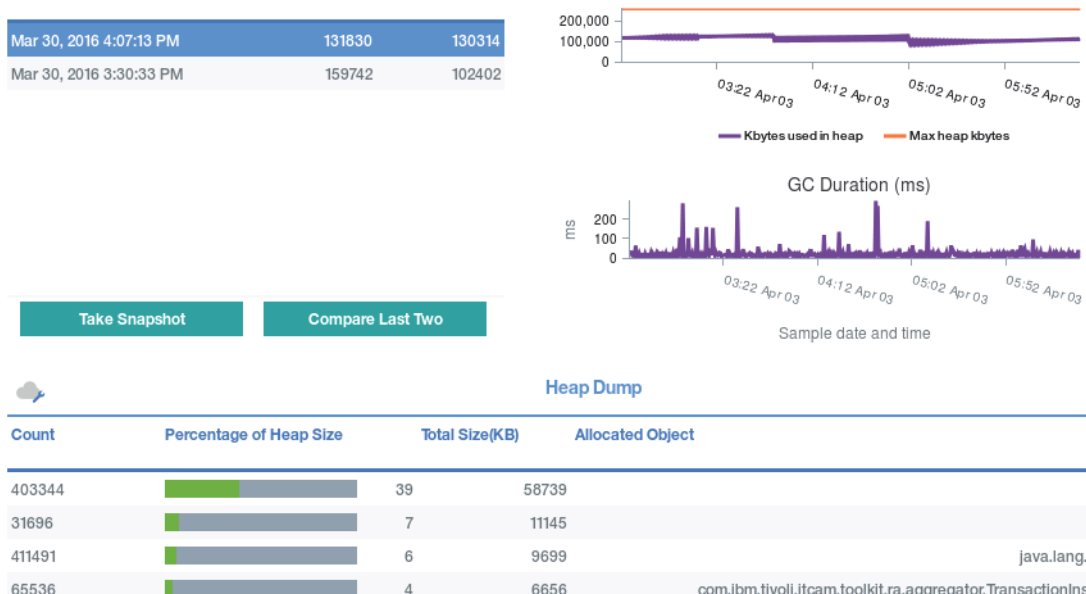
The **Requests with Slowest Response Time**, **In-Flight Request Summary**, and **Heap Dump** functions are covered in the next section.

## Exercise 2 Monitoring the heap

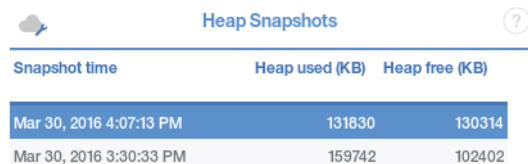
1. Click **View Heap Dump** on the **JVM GC (history)** widget.



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



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

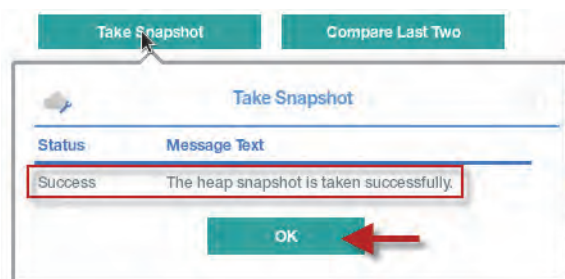


The screenshot shows a table titled "Heap Snapshots" with three columns: "Snapshot time", "Heap used (KB)", and "Heap free (KB)". There are two rows of data.

Snapshot time	Heap used (KB)	Heap free (KB)
Mar 30, 2016 4:07:13 PM	131830	130314
Mar 30, 2016 3:30:33 PM	159742	102402



4. Confirm successful creation of the second snapshot.

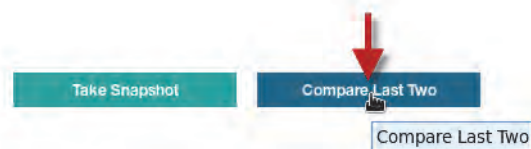


5. Click **OK** to complete the operation.
6. Click **Compare Last Two**.

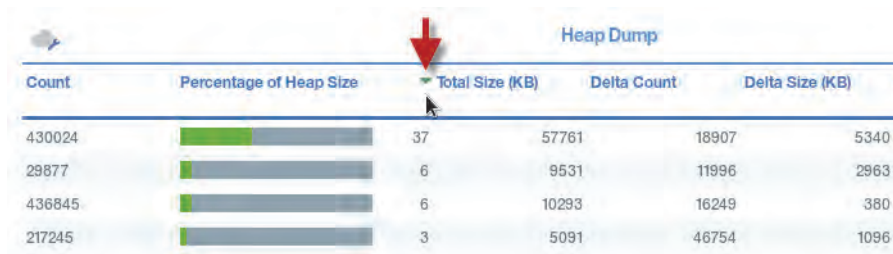


The screenshot shows a table titled "Heap Snapshots" with three columns: "Snapshot time", "Heap used (KB)", and "Heap free (KB)". There are three rows of data.

Snapshot time	Heap used (KB)	Heap free (KB)
Apr 3, 2016 6:42:30 AM	139688	122256
Mar 30, 2016 4:07:13 PM	131830	130314
Mar 30, 2016 3:30:33 PM	159742	102402



- 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.



Count	Percentage of Heap Size	Total Size (KB)	Delta Count	Delta Size (KB)	
430024	<div><div></div></div>	37	57761	18907	5340
29877	<div><div></div></div>	6	9531	11996	2963
436845	<div><div></div></div>	6	10293	16249	380
217245	<div><div></div></div>	3	5091	46754	1096

- Review the categories of data in the table.

Heap Dump

Count	Percentage of Heap Size	Total Size (KB)	Delta Count	Delta Size (KB)	Allocated Object	
430024	<div><div></div></div>	37	57761	18907	5340	char[]
29877	<div><div></div></div>	6	9531	11996	2963	byte[]
436845	<div><div></div></div>	6	10293	16249	380	java.lang.String
217245	<div><div></div></div>	3	5091	46754	1096	java.util.HashMap\$Entry
37439	<div><div></div></div>	2	3431	12073	831	java.util.HashMap\$Entry[]

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

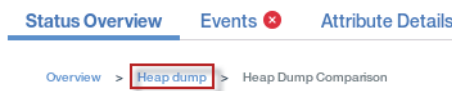
DELTA

Heap Dump

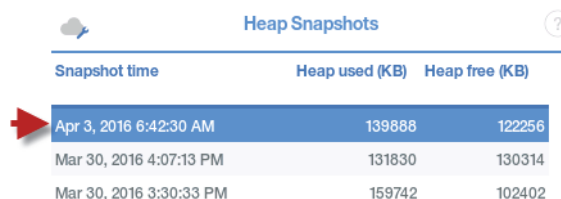
Delta Count	Delta Size (KB)	Allocated Object
57761	18907	5340char[]
9531	11996	2963byte[]
10293	16249	380java.lang.String

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

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

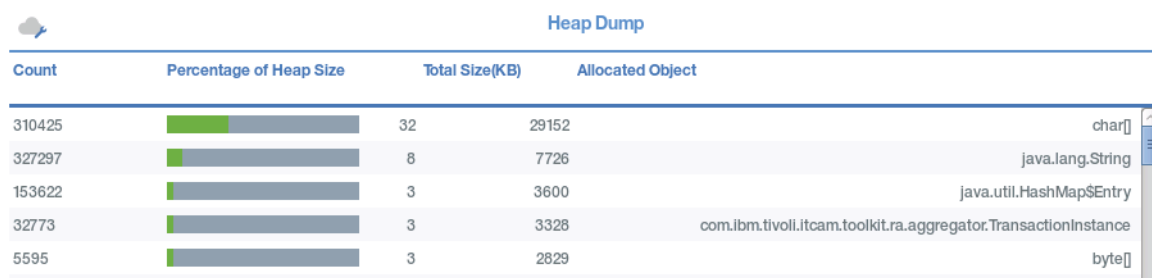


- Click a row in the **Heap Snapshot** widget.



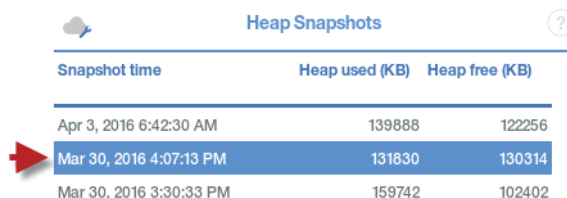
Snapshot time	Heap used (KB)	Heap free (KB)
Apr 3, 2016 6:42:30 AM	139888	122256
Mar 30, 2016 4:07:13 PM	131830	130314
Mar 30, 2016 3:30:33 PM	159742	102402

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



Count	Percentage of Heap Size	Total Size(KB)	Allocated Object
310425	<div><div></div></div>	32	29152
327297	<div><div></div></div>	8	7726
153622	<div><div></div></div>	3	3600
32773	<div><div></div></div>	3	3328
5595	<div><div></div></div>	3	2829

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



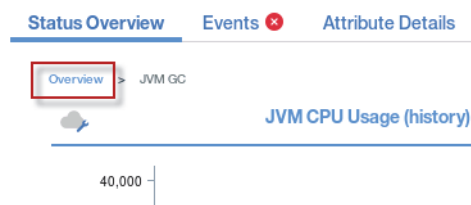
Snapshot time	Heap used (KB)	Heap free (KB)
Apr 3, 2016 6:42:30 AM	139888	122256
Mar 30, 2016 4:07:13 PM	131830	130314
Mar 30, 2016 3:30:33 PM	159742	102402

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



Count	Percentage of Heap Size	Total Size(KB)	Allocated Object
403344	<div><div></div></div>	39	58739
31696	<div><div></div></div>	7	11145
411491	<div><div></div></div>	6	9699
65536	<div><div></div></div>	4	6656

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



Status Overview Events Attribute Details

Overview > JVM GC

JVM CPU Usage (history)

40,000



## 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.

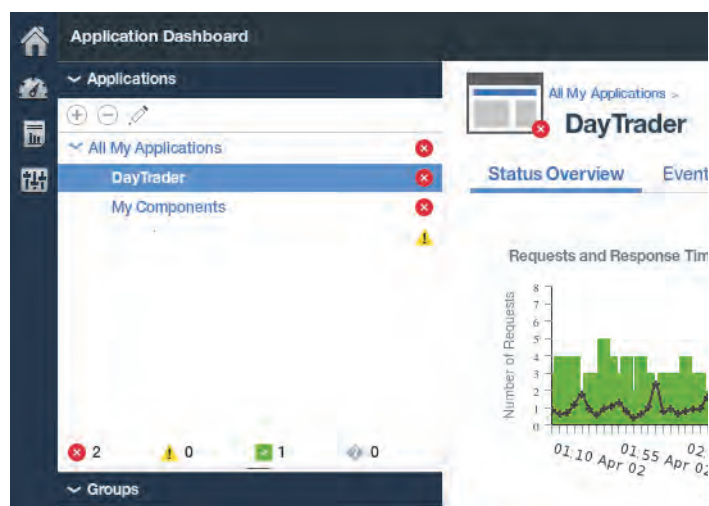
## Objectives

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

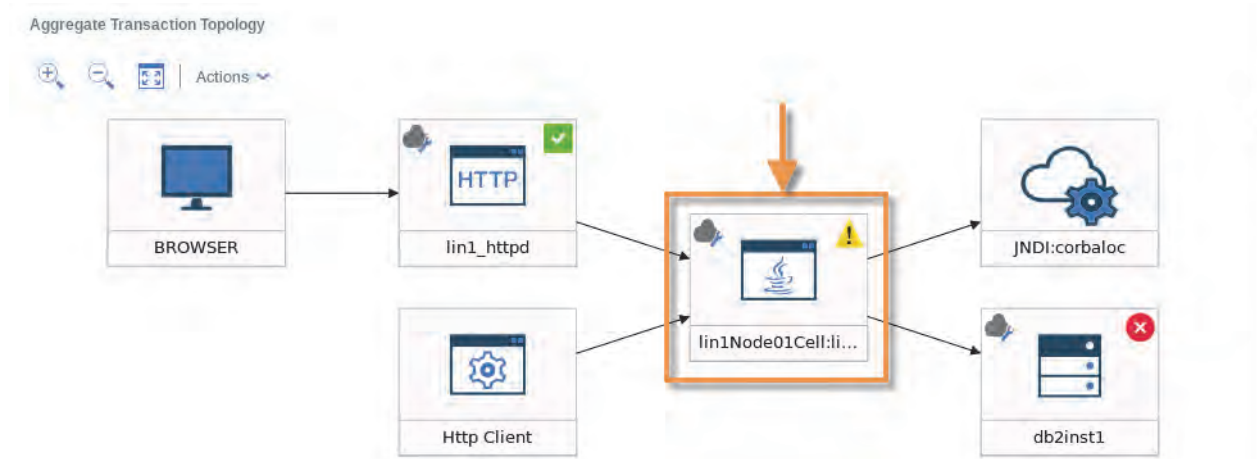
- Analyze request instances, method, and stack traces for WebSphere Application Server
- Analyze transaction topologies for WebSphere Application Server

## 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**.

Requests with Slowest Response Time

URL	Request Name	Average Response Time (ms)	Error Rate (%)
http://lin1.ibm.edu/daytrader/scenario	/daytrader/scenario	9	0.00%

Diagnose

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

Overview > Diagnostic Dashboard

**Request Summary**

Start record date and time: 2016-02-03 03:40:00 PM  
End record date and time: 2016-02-03 03:50:00 PM

Request Name	Average Respo...	Minimum Respo...	Maximum Respo...	Sampled Reque...	Request Count	CPU Time (ms)	Request Type	Action
/daytrader/scenario	20	1	78	39	56	16	Servlet	View instance data

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

Request Instances - /daytrader/scenario

Record Date And Time	Request Name	Request ID	Response Time (ms)	CPU Time (ms)	Request Type	Thread ID	Method Entries	Action
2016-04-09 12:20:00 PM	/daytrader/scenario	42949713330	74	45	Servlet	232	Yes	View request sequence
2016-04-09 12:20:00 PM	/daytrader/scenario	55834577...	46	16	Servlet	662	No	View request sequence
2016-04-09 12:30:00 PM	/daytrader/scenario	386547456...	42	14	Servlet	229	No	View request sequence
2016-04-09 12:20:00 PM	/daytrader/scenario	30064809...	40	12	Servlet	224	No	View request sequence
2016-04-09 12:25:00 PM	/daytrader/scenario	30064809...	12	9	Servlet	224	No	View request sequence
2016-04-09 12:30:00 PM	/daytrader/scenario	42949713372	10	8	Servlet	232	No	View request sequence
2016-04-09 12:25:00 PM	/daytrader/scenario	42949713353	8	6	Servlet	232	No	View request sequence
2016-04-09 12:25:00 PM	/daytrader/scenario	51539642506	8	6	Servlet	279	No	View request sequence
2016-04-09 12:30:00 PM	/daytrader/scenario	30064809...	8	6	Servlet	224	No	View request sequence









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

Request Sequence - 42949713330


Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)
+	1	1	/daytrader/scenario	2016-04-09 12:20:00 PM

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










Request Sequence - 42949713330

Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)	Event Type	CPU Time (ms)
	1	1 /daytrader/scenario	2016-04-09 12:20:00 PM	<div><div></div></div>	74 Servlet	45
	2	2 doGet	2016-04-09 12:20:00 PM	<div><div></div></div>	73 Method	44
	3	3 performTask	2016-04-09 12:20:00 PM	<div><div></div></div>	73 Method	44
	4	4 /daytrader/scenario	2016-04-09 12:20:00 PM	<div><div></div></div>	73 Servlet	44
	5	5 doGet	2016-04-09 12:20:00 PM	<div><div></div></div>	73 Method	43
	6	6 performTask	2016-04-09 12:20:00 PM	<div><div></div></div>	73 Method	43
	7	7 doHome	2016-04-09 12:20:00 PM	<div><div></div></div>	72 Method	43
<div></div>	8	8 /daytrader/scenario	2016-04-09 12:20:00 PM	<div><div></div><div></div></div>	37 Servlet	35

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



Request Sequence - 30064813502

Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)	Event Type	CPU Time (ms)
	1	1	/daytrader/scenario	2016-04-10 09:10:00 AM		43 Servlet
	2	2	/daytrader/scenario	2016-04-10 09:10:00 AM		42 Servlet
	3	3	/daytrader/scenario	2016-04-10 09:10:00 AM	 	35 Servlet
	4	4	/daytrader/scenario	2016-04-10 09:10:00 AM	 	35 Servlet

Request Context - /daytrader/scenario		Request Stack Trace - /daytrader/scenario	
Name	Value	Fully Qualified Method Name	
fullRequestName	DayTrader3-EE6#web.war/daytrader/s...		No items to display
requestType	Servlet		
requestName	/daytrader/scenario		

No items to display

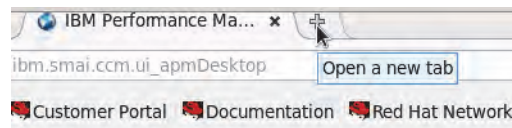


**Note:** Your requests might not have data in one of both of the request widgets.

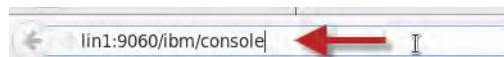
## Exercise 2 Generating more traffic

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.



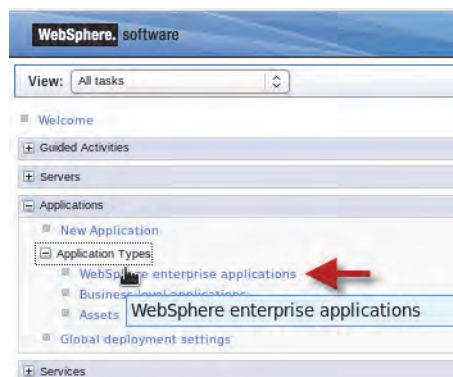
2. Enter **lin1:9060/ibm/console** in the address bar of your browser and press **Enter**.



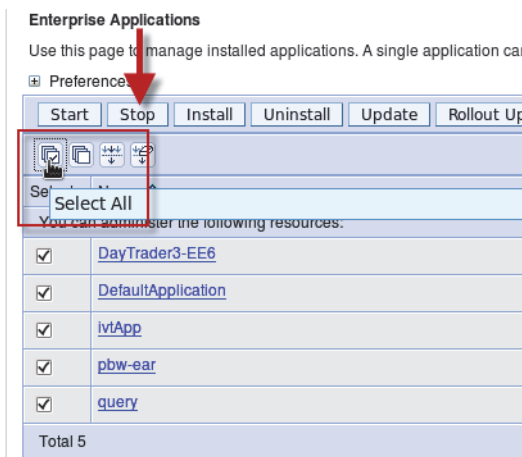
3. Enter the user ID **wasadmin** and click **Log in**.



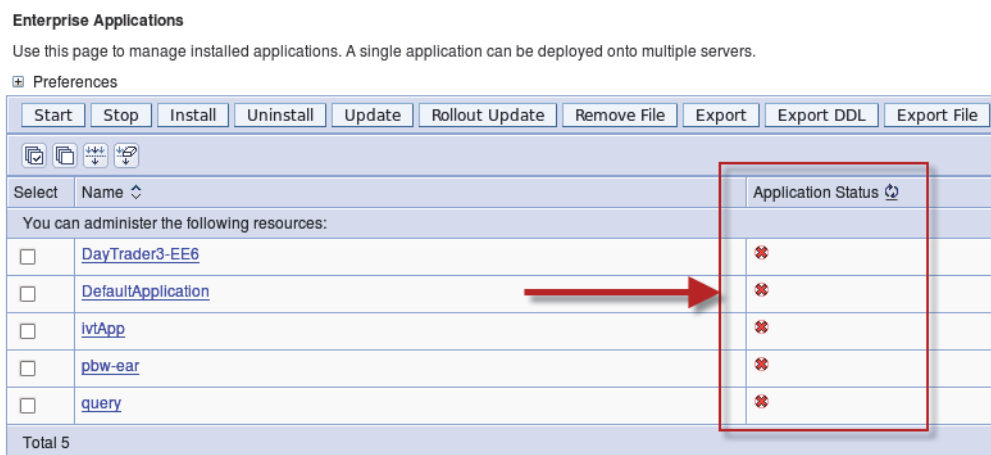
4. Navigate to **Applications > Application Types > WebSphere enterprise applications**.



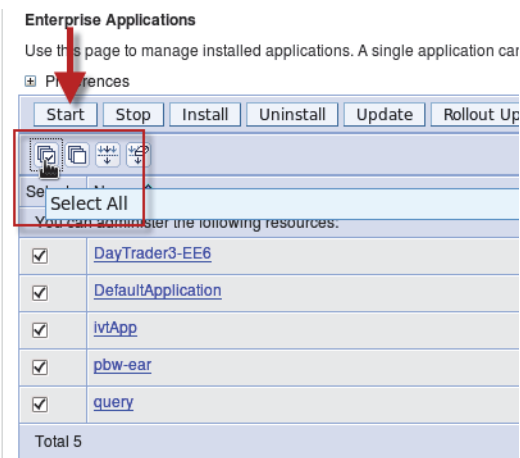
5. Click the **Select All** icon; click **Stop**.



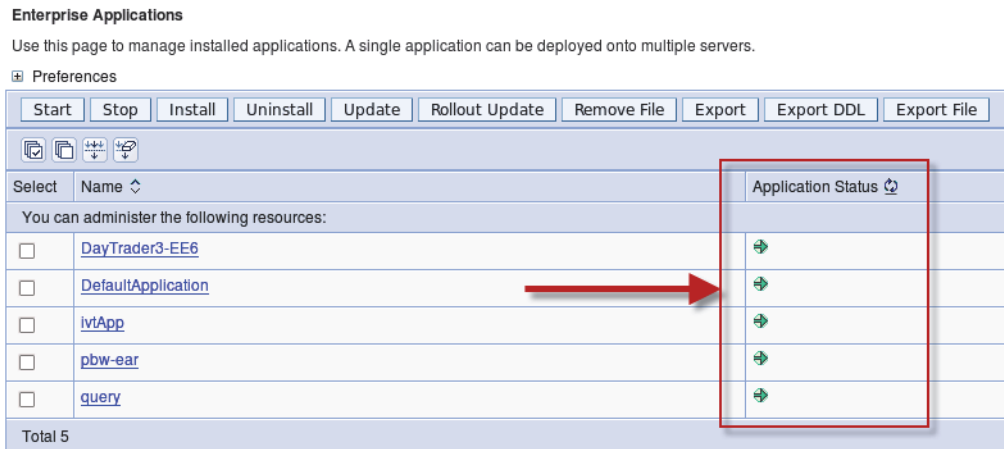
6. Confirm that the applications were stopped successfully.



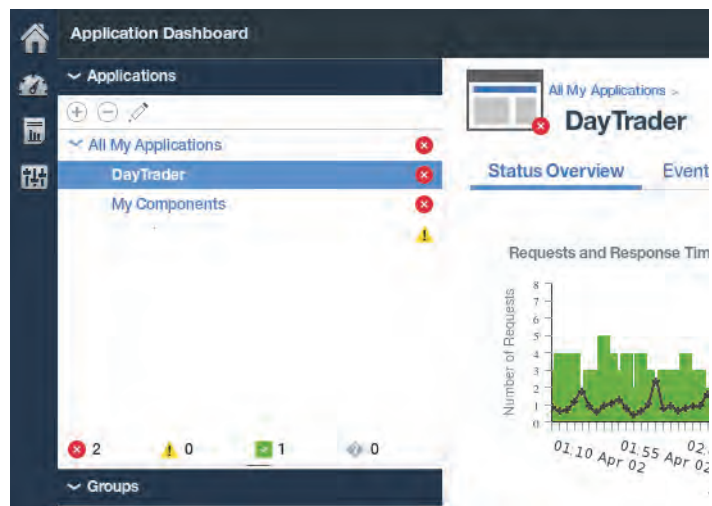
7. Click the **Select All** icon; click **Start**.



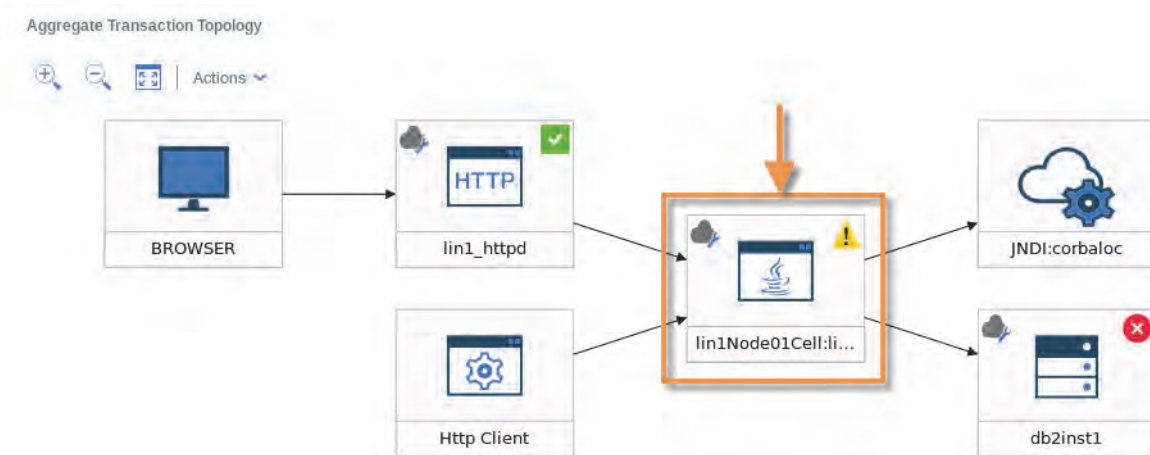
8. Confirm that the applications were started successfully.



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/**.

URL	Request Name	Average Response Time (ms)	Error Rate (%)
http://in1:9060/ibm/console/login.do	/ibm/console/login.do	670	0.00%
http://in1:9060/ibm/console/nsc.do	/ibm/console/nsc.do	147	0.00%
http://in1:9060/ibm/console/navigation.do	/ibm/console/navigation.do	45	0.00%
http://in1.ibm.edu/daytrader/scenario	/daytrader/scenario	43	0.00%
http://in1:9060/ibm/console/unsecureL...	/ibm/console/unsecureLogon.jsp	24	0.00%

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.

Record Name	Average Response Time (ms)	Minimum Response Time (ms)	Maximum Response Time (ms)	Sampled Request Count	Request Count	CPU Time (ms)	Request Type	Action
console/collectionBu...	7687	5561	10827	3	3	4489	Servlet	View instance data
console/login.do	2250	2250	2250	1	1	844	Servlet	View instance data
console/navigationC...	1047	1047	1047	1	1	601	Servlet	View instance data
console/nsc.do	147	147	147	1	1	147	Servlet	View instance data



**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.

Time And Time	Request Name	Request ID	Response Time (ms)	CPU Time (ms)	Request Type	Thread ID	Method Entries	Action
11:13:44 PM	/ibm/console/collection...	51539643198	10827	4340	Servlet	217	Yes	View request sequence
11:18:04 PM	/ibm/console/collection...	60129554882	9779	4323	Servlet	482	Yes	View request sequence
11:13:32 PM	/ibm/console/collection...	51539643178	6675	4942	Servlet	217	Yes	View request sequence

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

Request Sequence - 30064812999

Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)	Event Type	CPU Time (ms)
	10	getWscNavigationTree	2016-04-10 05:45:21 AM		52 Method	49
	10	callStaticFilters	2016-04-10 05:45:21 AM		383 Method	374
	15	initWSCConfig	2016-04-10 05:45:22 AM		158 Method	156
	9	init	2016-04-10 05:45:22 AM		86 Method	85
	10	lookup	2016-04-10 05:45:22 AM		83 JNDI	82
	9	loginInit	2016-04-10 05:45:22 AM		67 Method	67
	19	/ibm/console/secure/jsclite/tiles/frameSet.jsp	2016-04-10 05:45:22 AM		3 Servlet	3
	20	/ibm/console/secure/jsclite/tiles/frameSet.jsp	2016-04-10 05:45:22 AM		1 Servlet	1

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.

Request Sequence - 30064812999

Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)	Event Type	CPU Time (ms)
	10	getWscNavigationTree	2016-04-10 05:45:21 AM		52 Method	49
	10	callStaticFilters	2016-04-10 05:45:21 AM		383 Method	374
	15	initWSCConfig	2016-04-10 05:45:22 AM		158 Method	156
	9	init	2016-04-10 05:45:22 AM		86 Method	85
	10	lookup	2016-04-10 05:45:22 AM		83 JNDI	82
	9	loginInit	2016-04-10 05:45:22 AM		67 Method	67
	19	/ibm/console/secure/jsclite/tiles/frameSet.jsp	2016-04-10 05:45:22 AM		3 Servlet	3
	20	/ibm/console/secure/jsclite/tiles/frameSet.jsp	2016-04-10 05:45:22 AM		1 Servlet	1

Request Context - lookup

Name	Value
lookupString	com.ibm.isc.PluginRegistry/lin1Node01
url	corbaloc:rir/NameServiceServerRoot
requestType	JNDI

Request Stack Trace - lookup

Fully Qualified Method Name
javax.naming.InitialContext.lookup(436:InitialContext.java)
com.ibm.wsspi.RegistryLoader.getPluginRegistry(226:RegistryLoader.java)
com.ibm.wsspi.IPluginRegistryFactory.getPluginRegistry(39:IPluginRegistryFa...

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

Request Stack Trace - /daytrader/scenario

Fully Qualified Method Name	Online help







## 4 Transaction tracking

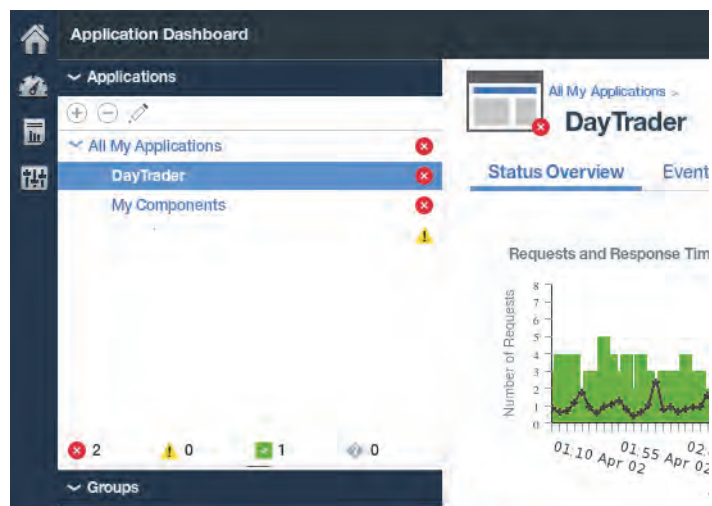
In addition to resource monitoring and code-level diagnosis, IBM Application Performance Management Advanced Diagnostics supports transaction tracking.

### Objectives

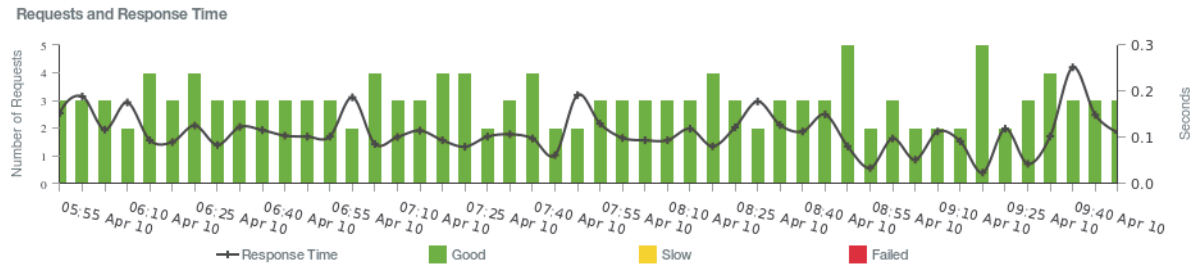
After completing all the exercises, you can analyze transaction topologies for WebSphere Application Server.

## Exercise 1 Exploring aggregate topologies

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

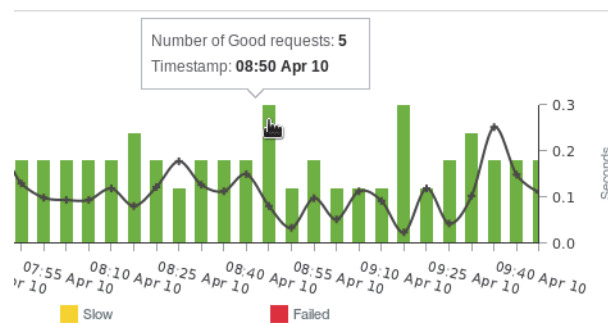


2. Review the chart for top-level **Requests and Response Time** widget.

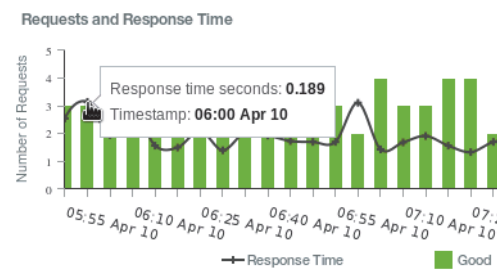


The bars indicate the number of requests for a specific collection period. The three colors are for Good, Slow, and Failed requests. The single line at this level graphs response time.

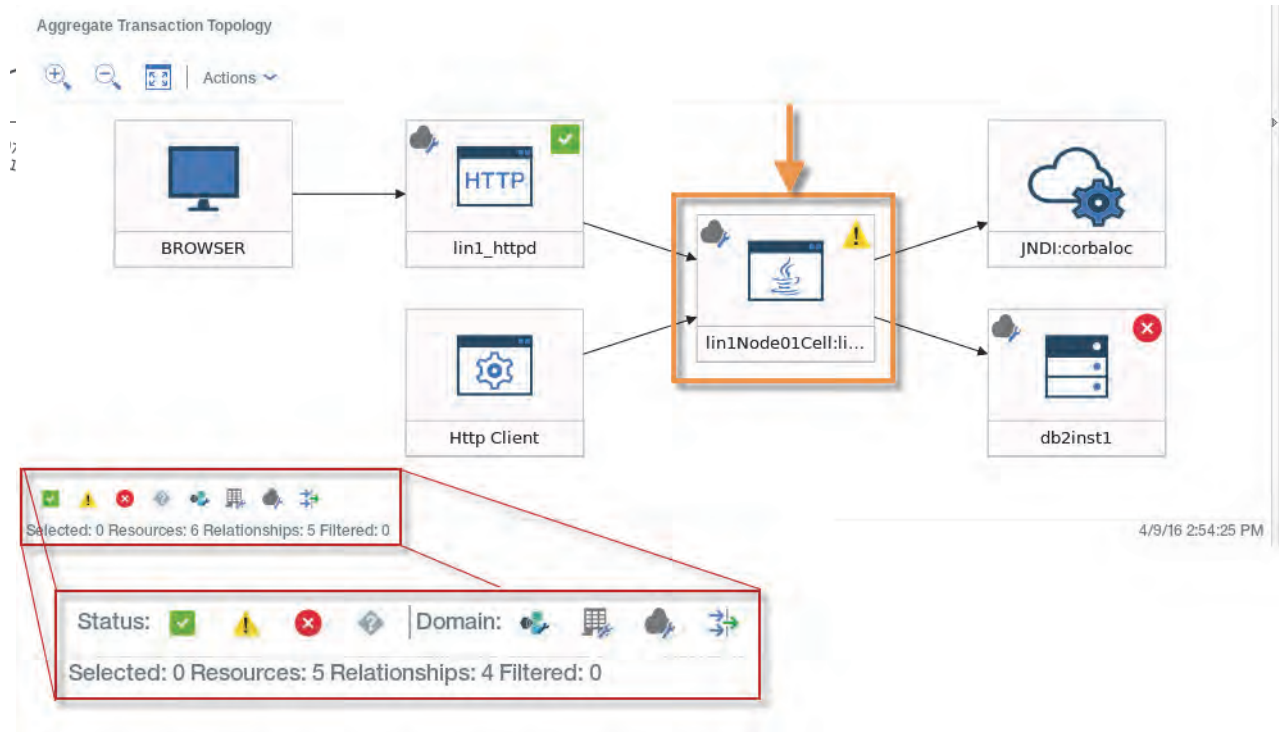
3. Position the cursor over one of the bars, and note the request and time information.



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.



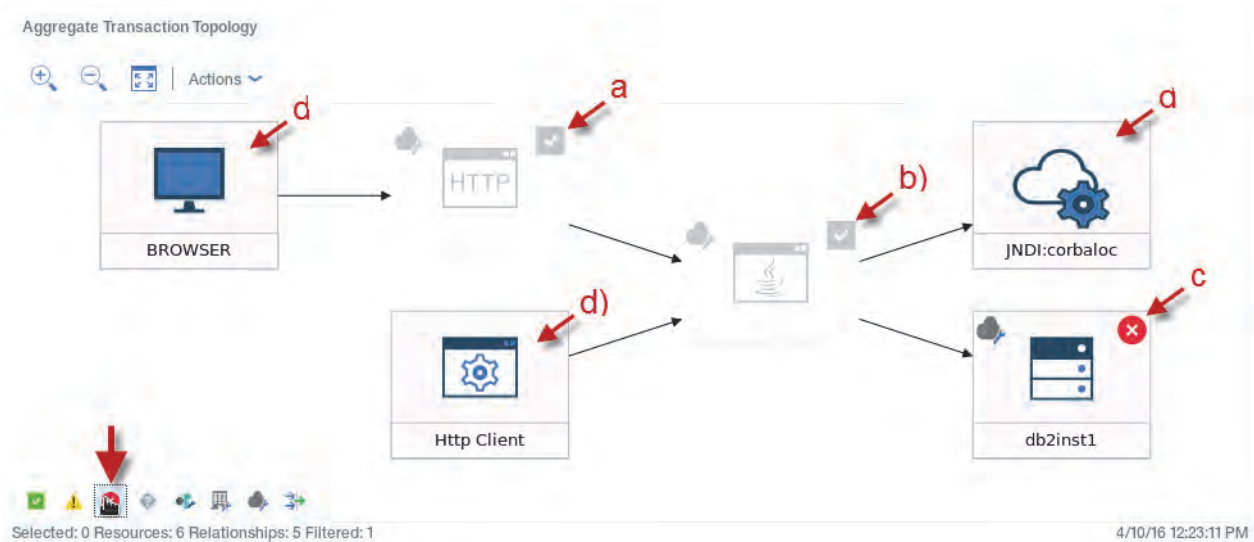
5. Review this sample **Aggregate Transaction Topology**, including the icons at the lower left.



**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.

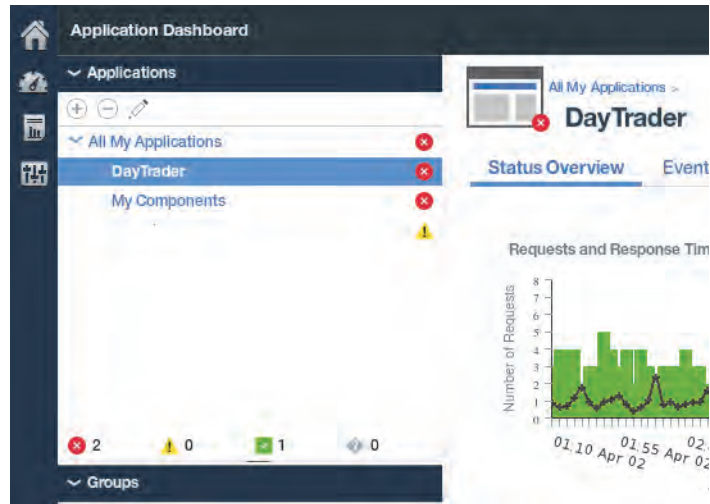
6. Note the changes in the **Aggregate Transaction Topology** when the critical icon is selected.



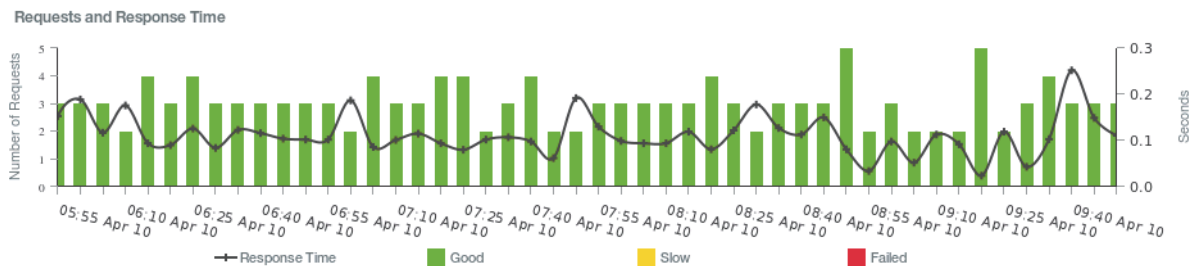
- a. The **lin1 HTTP** node previously had a normal status, which was indicated by a white check mark in a green box (see the screen capture in [Step 5](#) on page 39). The node is now disabled **for display purposes only**.
  - b. The **lin1Node01Cell** previously had a normal status, which is indicated 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

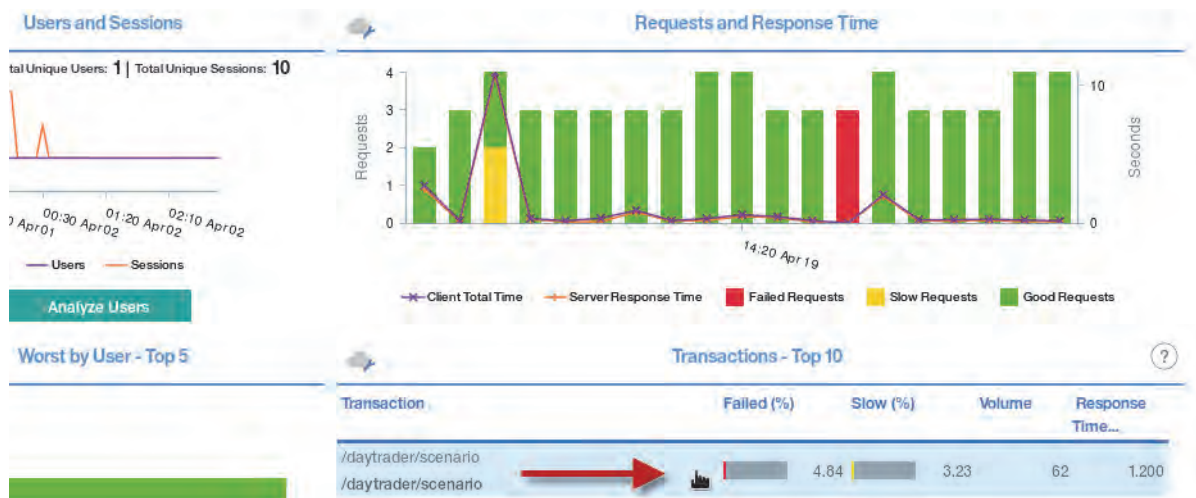
1. Return to the top-level **Status Overview** dashboard for the **DayTrader** application.



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

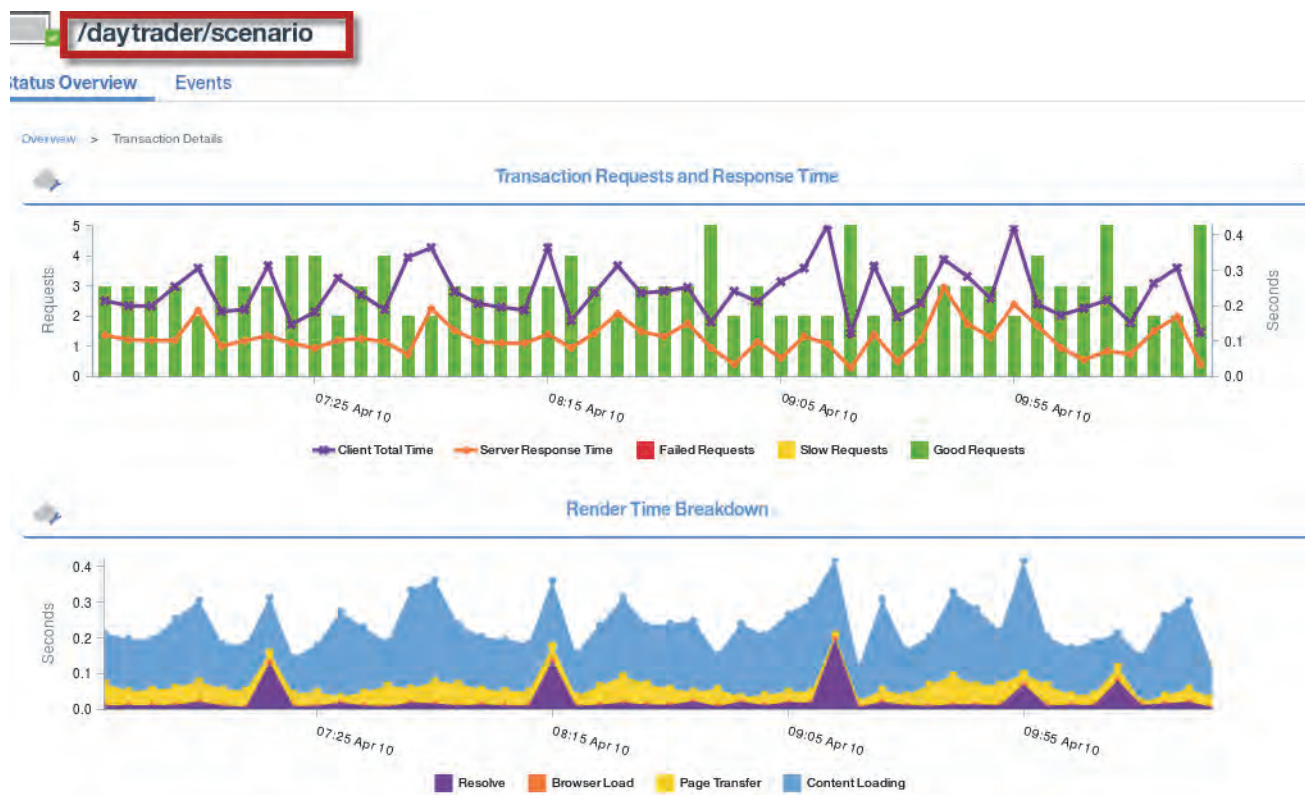


3. Locate and click the **/daytrader/scenario** transaction in the **Transactions Top 10** list.



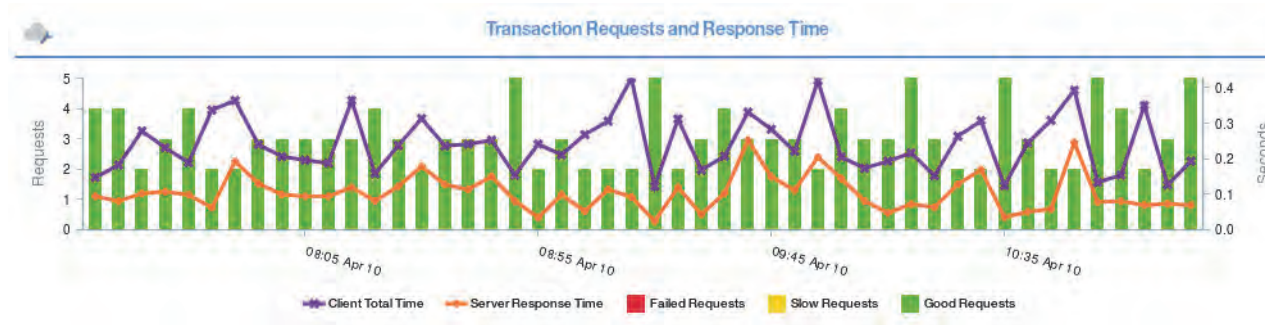


4. Review the two graphs at the top of the **/daytrader/scenario** dashboard.



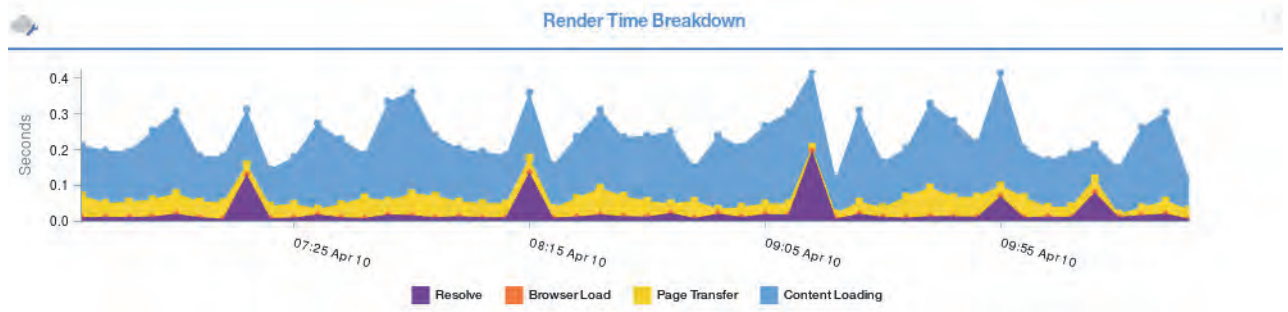
The **Transaction Requests and Response Time** graph is similar to the graph at [Step 2](#) on page 41.

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**.

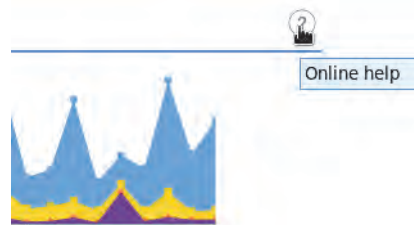


**Note:** The **Client Total Time** in this second-level graph corresponds to the **Response timeline** in the top-level graph.

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.

KPI	Description	Derived from
Response Time	<p>The time taken, in seconds, for the following metrics within the web page to complete. Timing points are described by the <a href="#">W3C Navigation Timing</a> specification.</p> <ul style="list-style-type: none"><li>• Resolve - the time taken for the DNS lookup and any page redirections (navigationStart - domainLookupEnd)</li><li>• Page Transfer - the time taken to download the original page and transfer HTTP and TCP data (connectStart -</li></ul>	BROWSETIME

9. Click the X to close the help dialog.

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

Runs on			Subtransactions - /daytrader/scenario				
Web Servers	Failed (%)	Slow (%)	Transaction	Type	Failed (%)	Slow (%)	Volume
192.168.1.104:80	<div></div> 1.38	<div></div> 0.92	No items to display				

Transaction Instances					
Timestamp	Status	Response Time (s)	Source	User Name	User Agent
Apr 21, 2016, 05:55	✓	0.220	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Apr 21, 2016, 05:55	✓	0.288	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Apr 21, 2016, 05:50	✓	0.303	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Apr 21, 2016, 05:50	✓	0.314	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0



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

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

#### Transaction Instances

Timestamp	Status	Response Time (Seconds)	Source	User Name	User Agent
Sep 12, 2015, 19:37	✓	0.625	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 ...

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.

Transaction Instances					
Timestamp	Status	Response Time (s)	Source	User Name	User Agent
Apr 28, 2016, 01:50	✓	0.314	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Apr 28, 2016, 01:50	✓	0.020	192.168.1.104	Anonymous	Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0

13. Click the top transaction in the list.

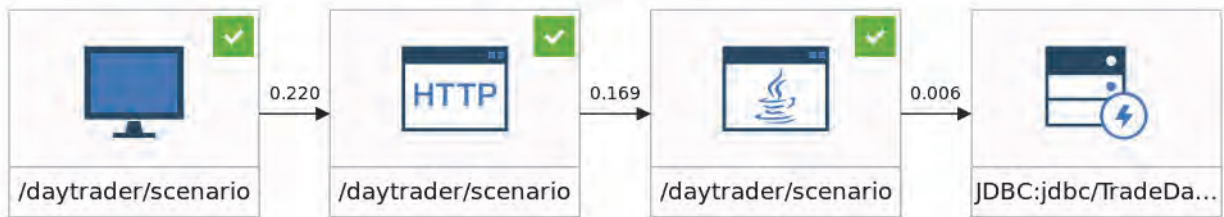


14. Review the topology.

Transactions - /daytrader/scenario

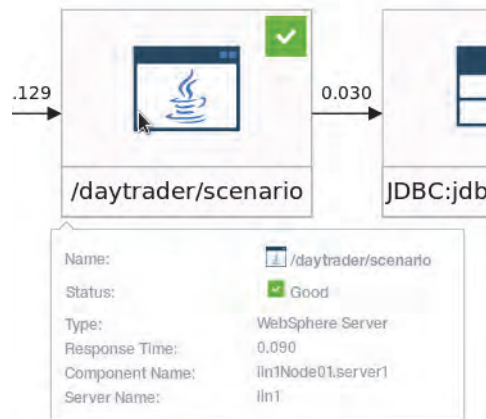
Timestamp	Status	Response Time (sec)	User Name
Apr 21, 2016, 05:55	Good	0.220	Anonymous

Actions



**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.

## 4 Transaction tracking

### Exercise 2 Exploring transaction instance topologies

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

The screenshot shows the 'Node Properties' widget with the following data:

Node Properties		WAS Status	
Timestamp	Apr 28, 2016, 01:40	Server status	Connected
Transaction	/daytrader/scenario	Slowest response time (ms)	1,965
Response Time(s)	0.090	JVM memory used (KB)	112,247
Component Name	lin1Node01.server1	Errors in log	0
Component Type	WebSphere Application Se...	JVM memory total (KB)	184,704
Host Name	lin1	Warnings in log	0
Server Status	Good	Heap free after GC(%)	0.00%
Diagnose		JVM CPU used (%)	0.00%

Below the Node Properties widget, the 'Linux OS' section shows various system metrics:

- Online logical processors: 8 (red) / 4 (green)
- Aggregate CPU usage (%): 0 to 100
- Memory usage (%): 0 to 100
- Total disk usage (%): 0 to 100
- Network usage (Pkts/sec): 0 to 100

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

The screenshot shows the 'Request Sequence' dashboard for the transaction '/daytrader/scenario'. The dashboard displays a list of events with the following columns: Order ID, Depth, Event Name, Start Date And Time, Response Time (ms), Event Type, and CPU Time (ms).

Order ID	Depth	Event Name	Start Date And Time	Response Time (ms)	Event Type	CPU Time (ms)
2	2	doGet	2016-04-28 01:40:00 AM	89	Method	58
3	3	performTask	2016-04-28 01:40:00 AM	89	Method	58
4	4	/daytrader/scenario	2016-04-28 01:40:00 AM	89	Servlet	58
5	5	doGet	2016-04-28 01:40:00 AM	88	Method	58
6	6	performTask	2016-04-28 01:40:00 AM	88	Method	58
7	7	doHome	2016-04-28 01:40:00 AM	88	Method	57
8	8	/daytrader/scenario	2016-04-28 01:40:00 AM	48	Servlet	45
9	9	/daytrader/scenario	2016-04-28 01:40:00 AM	45	Servlet	44

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

8	8	/daytrader/scenario	2016-04-28 01:40:00 AM	48	Servlet	45
	9	/daytrader/scenario	2016-04-28 01:40:00 AM	45	Servlet	44

Request Context - /daytrader/scenario

Request Stack Trace - /daytrader/scenario

Name	Value	Fully Qualified Method Name
fullRequestName	DayTrader3-EE6#web.war/daytrader/s...	No items to display
requestType	Servlet	
requestName	/daytrader/scenario	

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

Status Overview

Events

Overview

Transaction Details

Transaction Topology

Request Sequence Dashboard

Transaction Topology

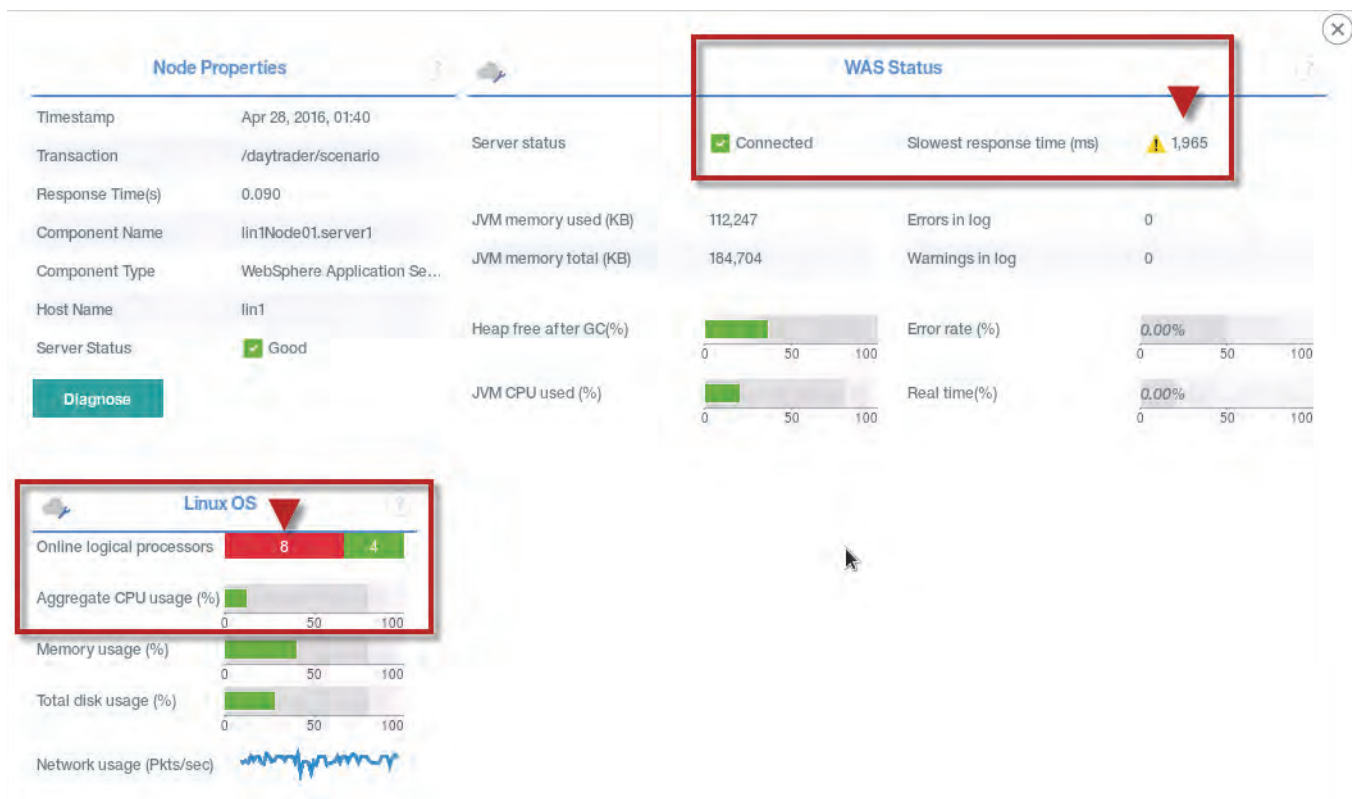
Request Sequence - 34359740172

Order ID	Depth	Event Name	Start Date And Time	Respo
2	2	doGet	2016-04-28 01:40:00 AM	

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

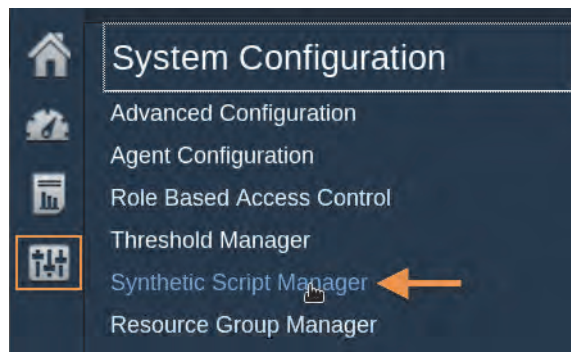
In these exercises, you create a synthetic transaction for Playback and monitoring. By completing these tasks, you enable periodic monitoring of the website that is accessed by the script.

### Objectives

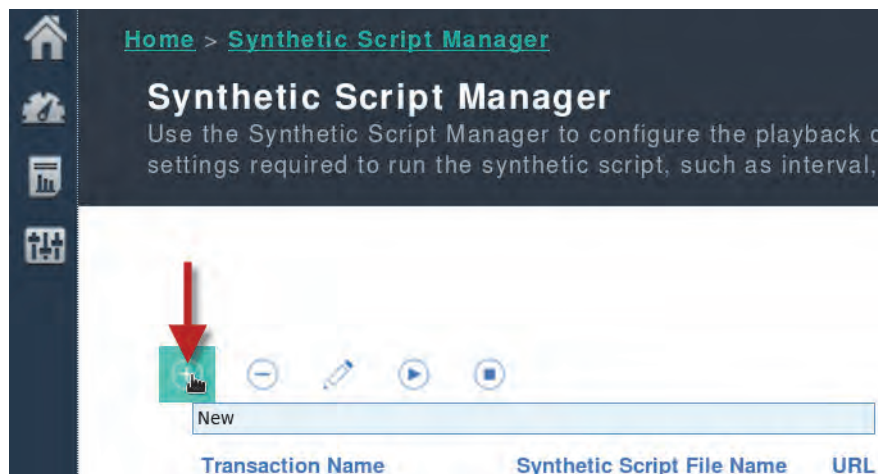
After completing all exercises, you can create and monitor synthetic transactions.

## Exercise 1 Configuring a synthetic transaction

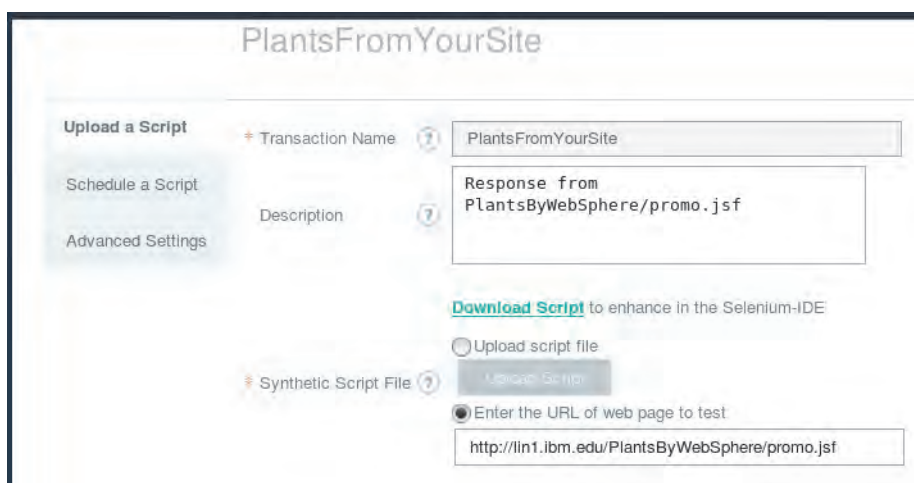
1. In the Performance Management console, hover over the sliders icon to bring up the **System Configuration** menu; then, select **Synthetic Script Manager**.



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**

PlantsFromYourSite

Upload a Script

Schedule a Script

Advanced Settings

Playback Mode: ☒ Simultaneous ☐ Staggered

Interval: 5 Minutes

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.

Transaction Name	Synthetic Script File Name	URL	Description	Locations
PlantsFromYourSite	load_lin1.ibm.edu.html	http://lin1.ibm.edu/Pla	Response from PlantsByWebSp	YourSite

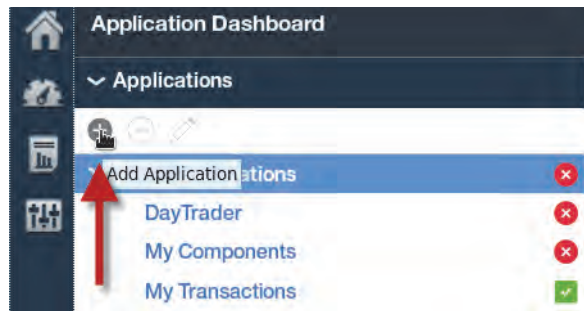
Description	Locations	Status	Playback Mode	Interval	Measureme Month
Response from PlantsByWebSp	YourSite	Started	simultaneous	5	

## 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.

A screenshot of the 'Add Application' dialog box. It has a dark blue header with 'Cancel' and 'Save' buttons. The main area contains a form with 'Application name \*' (required), 'Enter a unique name', and 'Description' fields. A 'Read...' button is on the right. The 'Application name' field is currently empty.

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

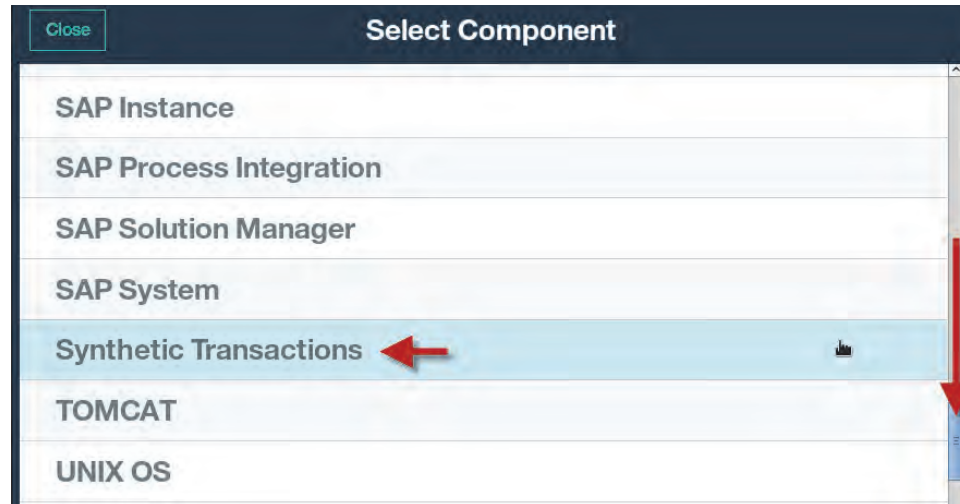
A screenshot of the 'Add Application' dialog box, similar to the previous one, but with the text 'PlantsFromYourSite' entered into the 'Application name' field. The 'Description' field is still empty.

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

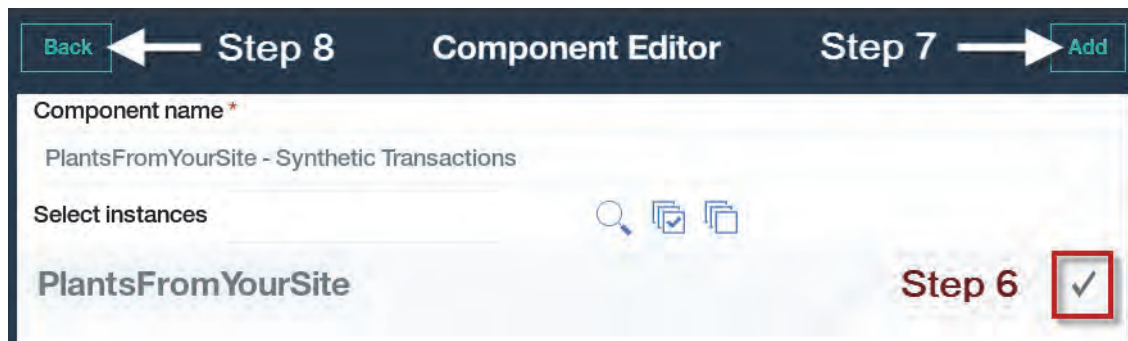
A screenshot of the 'Add components' dialog box. It has a dark blue header with 'Cancel' and 'Add components' buttons. The main area shows a 'Template \*' dropdown menu with 'Custom Application' selected. Below it is an 'Application components' section. A green plus icon is visible in the bottom right corner.



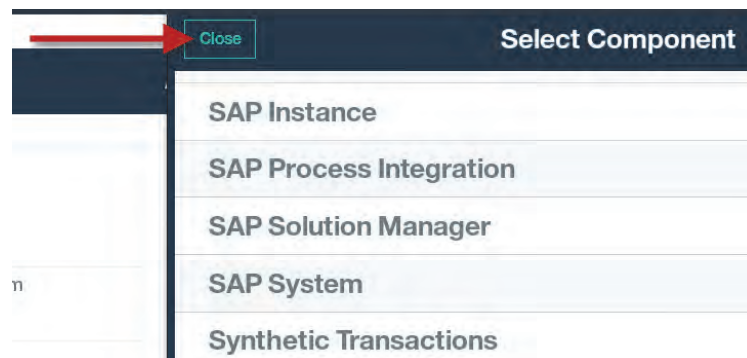
5. Scroll down to **Synthetic Transactions** in the **Select Component** dialog box and click it.



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.

Cancel Add Application Step 11 → Save

Application name \*  
PlantsFromYourSite Read...

Description

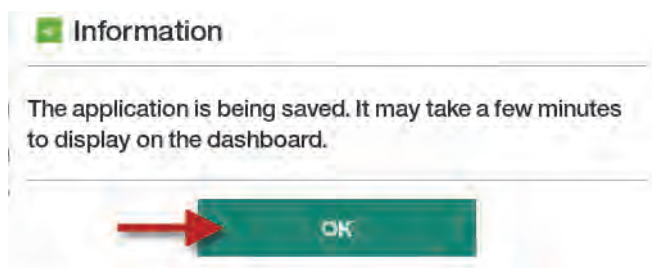
Application read from

Template \*  
Custom Application

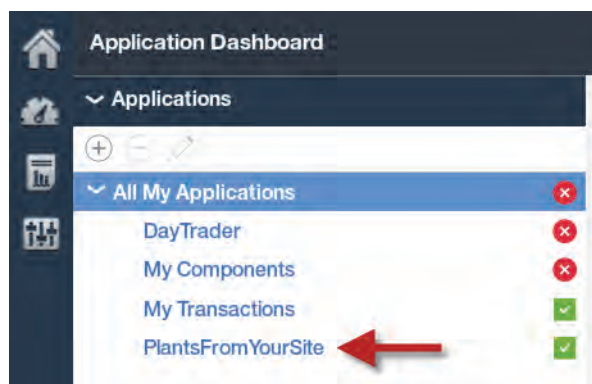
Application components  
PlantsFromYourSite - Synthetic Transactions(1)  
PlantsFromYourSite ← Step 10

11. Click **Save**.

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

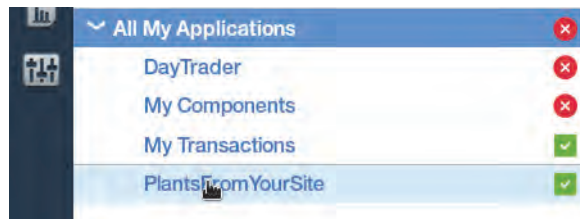


You are returned to the Application Dashboard and the **PlantsFromYourSite** application is displayed.



## Exercise 3 Viewing synthetic transactions

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



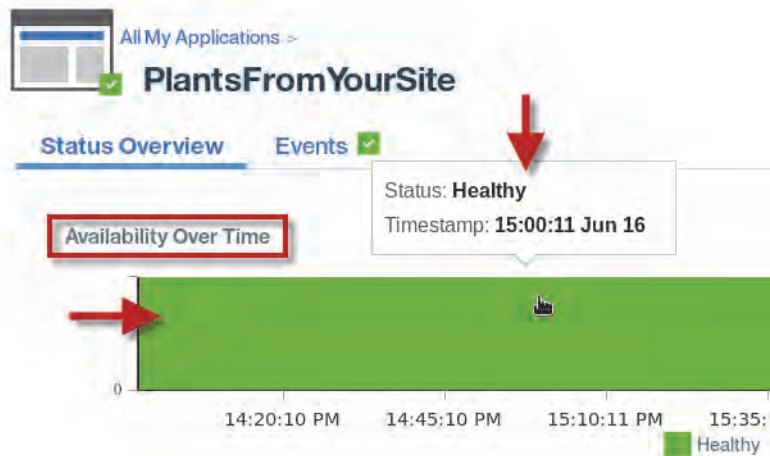
2. Synthetic transaction data is first displayed in the Availability Over Time widget.



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.

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



- Click the **Availability Over Time** widget.  
The **Synthetic Transactions** page is displayed.

Transaction List					
Transaction	Latest Status	Last Run At	Response Time Threshold (Seconds)	Latest Response Time (Seconds)	Average Response Time (Seconds)
PlantsFromYourSite	✓	Jun 16, 2016, 18:00	10.0	1.73	1.73
Total 1					

Location List				
Location	Latest Status	Last Run At	Latest Response Time (Seconds)	Average Response Time (Seconds)
YourSite	✓	Jun 16, 2016, 18:00	1.73	1.73

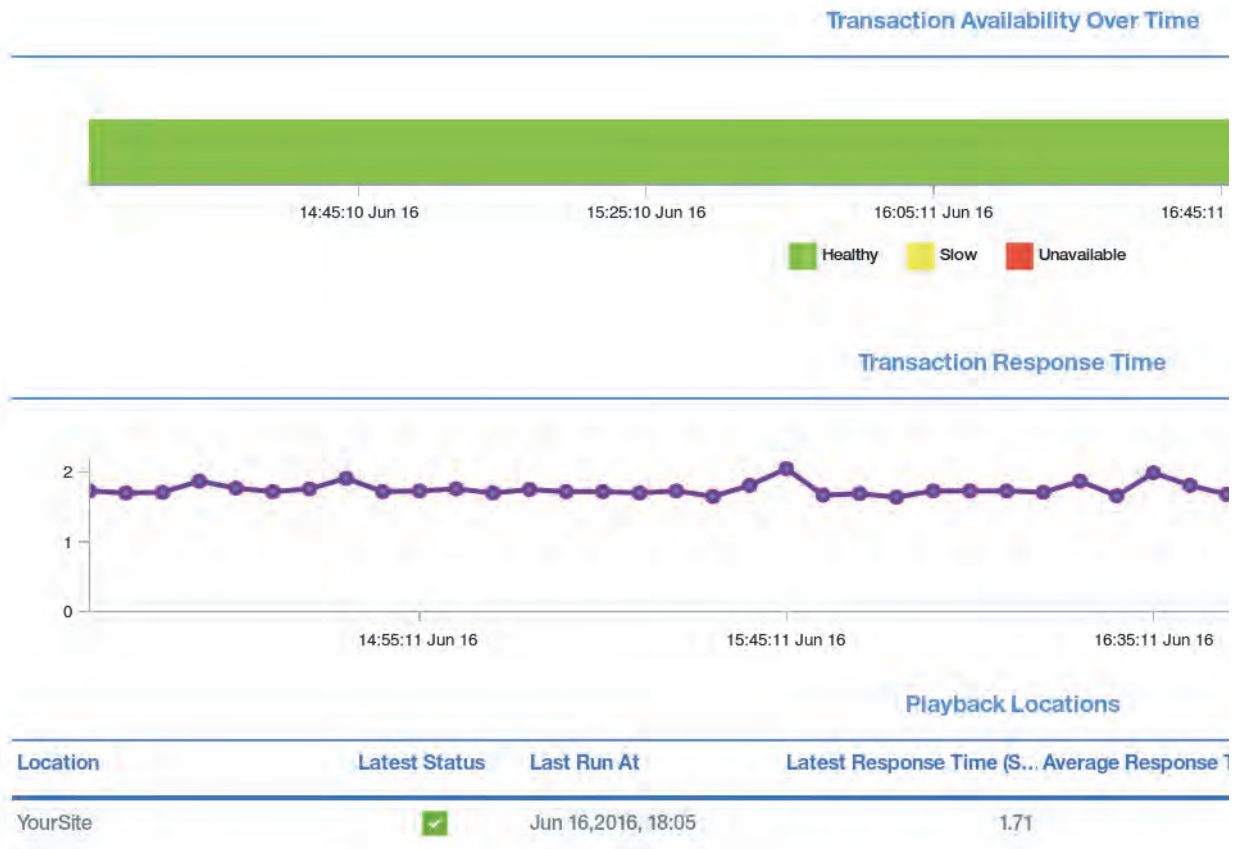
Examine the performance data for the transactions within Transactions List to determine whether any are having performance problems. When the same script is run from multiple locations, compare performance by location in the Location List. When multiple scripts are run from the same location, compare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

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

Transaction List				
Transaction	Latest Status	Last Run At	Response Time Threshold (S...)	Latest Response Time (S...
PlantsFromYourSite	✓	Jun 16, 2016, 18:00	10.0	1.73

The Synthetic Transaction page opens.

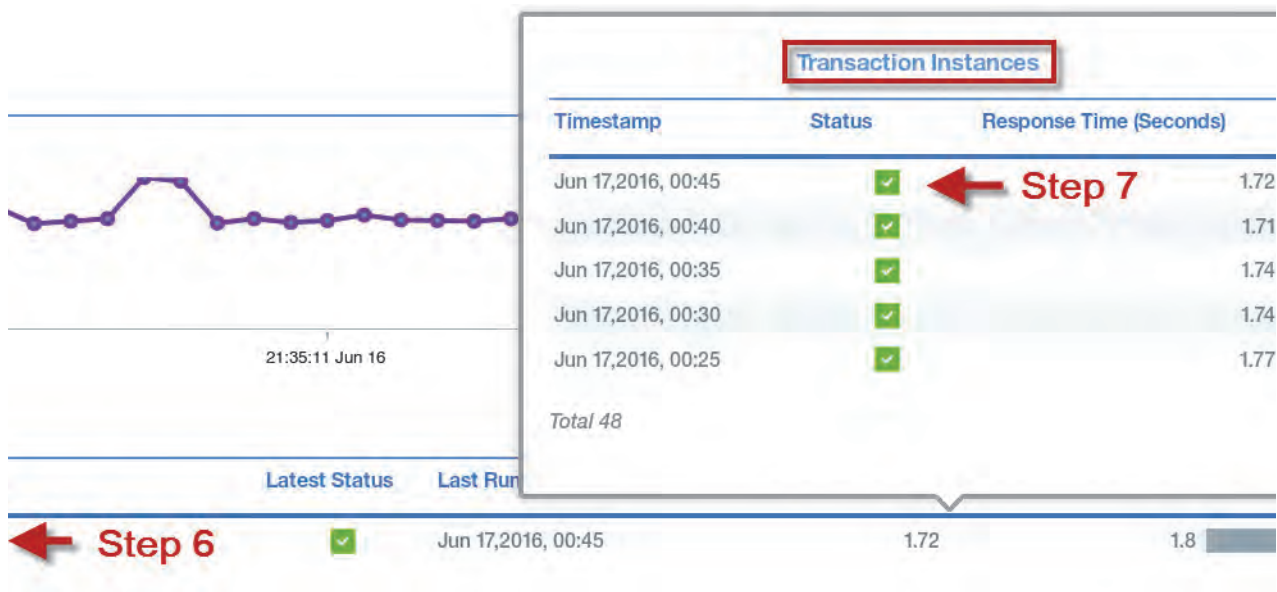


This page shows performance and availability data of this specific script over time.

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



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



- Click a **Transaction Instance** in the popup.
- Click any plus (+) sign to expand the **Transaction Break Down**.

Transaction Break Down							
Transaction Name	Response Time (Seconds)	Status	Http Status	Size (kb)	Blocked (ms)	D (n)	
PlantsFromYourSite	1.72	✓		0	0		
http://lin1.ibm.edu/Pla	0.4	✓		0	0		
GET:http://lin1.ibm	0.033		200 OK	14.52	0		

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

Transaction Break Down								
Transaction Name	Response Time (Seconds)	Status	Http Status	Size (kb)	Blocked (ms)	DNS (ms)	Connect (ms)	SSL (ms)
GET:http://lin1.ibm	0.005		200 OK	1.72	0	0	0	0
GET:http://lin1.ibm	0.02		200 OK	141.51	0	0	0	0
GET:http://lin1								
GET:http://lin1.ibm.edu/PlantsByWebSphere/javax.faces.resource/theme_summer1.gif.jsf?ln=images							5	0

- 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.

**Transaction Break Down** **Step 12** →

tp atus	Size (kb)	Blocked (ms)	DNS (ms)	Connect (ms)	SSL (ms)	Send (ms)	Wait (ms)	Receive (ms)
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
0 OK	14.52	0	0	0	0	0	30	0

**Step 11** ↖

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

Transaction Break Down

This group widget displays details of the response time of subtransactions (commands), and indicates failed subtransactions.

Various key performance indicators (KPIs) are supported by the Performance Management console.

**Transaction Name**  
The name of the subtransaction.

**Response Time (Seconds)**  
Total time in seconds for the transaction or subtransaction playback.

**Status**  
The transaction or subtransaction playback instance succeeded (Normal), was slow (Warning), or failed (Critical).

**Blocked Time (ms)**  
The blocked time of each entry.

**DNS Time (ms)**







## 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.

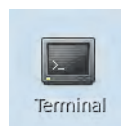
The Node.js agent is a single instance agent. It registers subnodes for each monitored Node.js application.

### 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.



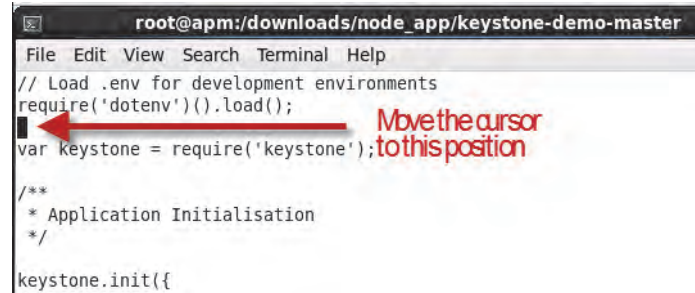
2. Navigate to the directory where the **keystone.js** file is located.

```
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.



```

root@apm:downloads/node_app/keystone-demo-master
File Edit View Search Terminal Help
// Load .env for development environments
require('dotenv').load();
var keystone = require('keystone');

/**
 * Application Initialisation
 */
keystone.init({

```

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');
```

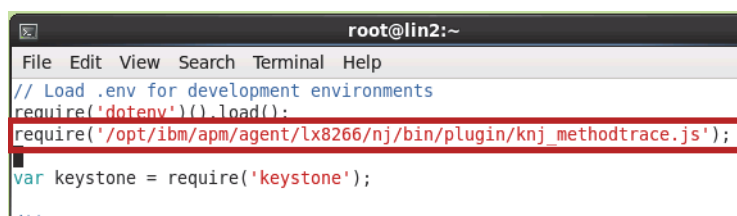
6. Type the following Require statement.

```
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.



```

root@lin2:~
File Edit View Search Terminal Help
// Load .env for development environments
require('dotenv').load();
require(' /opt/ibm/apm/agent/lx8266/nj/bin/plugin/knj_methodtrace.js');
var keystone = require('keystone');

```

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

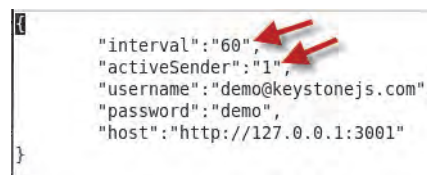
1. Type the following command to navigate to the directory containing the agent installation packages.

```
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.



```
{  
  "interval": "60",  
  "activeSender": "1",  
  "username": "demo@keystonejs.com",  
  "password": "demo",  
  "host": "http://127.0.0.1:3001"  
}
```

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/signin',  
                        '/keystone/posts/', '/keystone/galleries', '/keystone/enquiries', '/keystone/users',  
                        '/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.

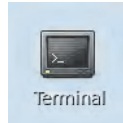
7. Type the following command to exit the file and make no changes.

```
: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.

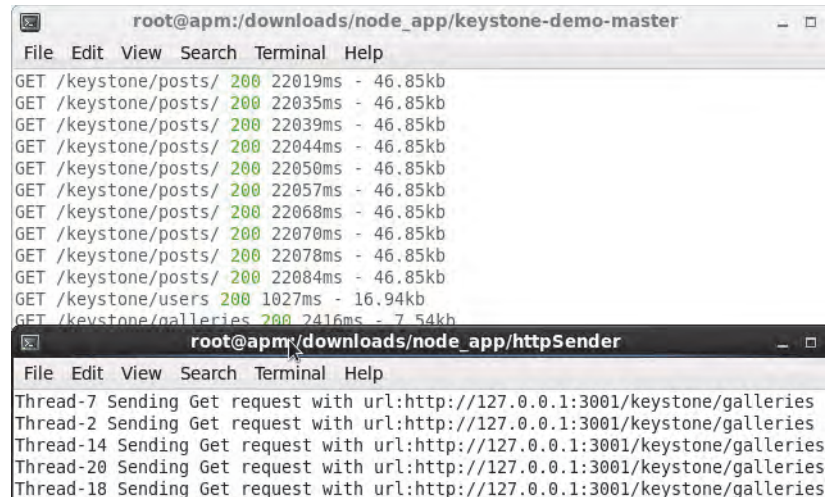
7. Open a separate terminal window, and type the command to navigate to the folder containing the traffic generation script. Press Enter.

```
cd /downloads/node_app/httpSender/
```

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

```
./sendUrlWithCookies.py
```

9. Review the flow of traffic.



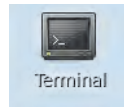
```
root@apm:/downloads/node_app/keystone-demo-master
File Edit View Search Terminal Help
GET /keystone/posts/ 200 22019ms - 46.85kb
GET /keystone/posts/ 200 22035ms - 46.85kb
GET /keystone/posts/ 200 22039ms - 46.85kb
GET /keystone/posts/ 200 22044ms - 46.85kb
GET /keystone/posts/ 200 22050ms - 46.85kb
GET /keystone/posts/ 200 22057ms - 46.85kb
GET /keystone/posts/ 200 22068ms - 46.85kb
GET /keystone/posts/ 200 22070ms - 46.85kb
GET /keystone/posts/ 200 22078ms - 46.85kb
GET /keystone/posts/ 200 22084ms - 46.85kb
GET /keystone/users 200 1027ms - 16.94kb
GET /keystone/galleries 200 2416ms - 7.54kb

root@apm:/downloads/node_app/httpSender
File Edit View Search Terminal Help
Thread-7 Sending Get request with url:http://127.0.0.1:3001/keystone/galleries
Thread-2 Sending Get request with url:http://127.0.0.1:3001/keystone/galleries
Thread-14 Sending Get request with url:http://127.0.0.1:3001/keystone/galleries
Thread-20 Sending Get request with url:http://127.0.0.1:3001/keystone/galleries
Thread-18 Sending Get request with url:http://127.0.0.1:3001/keystone/galleries
```

## 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

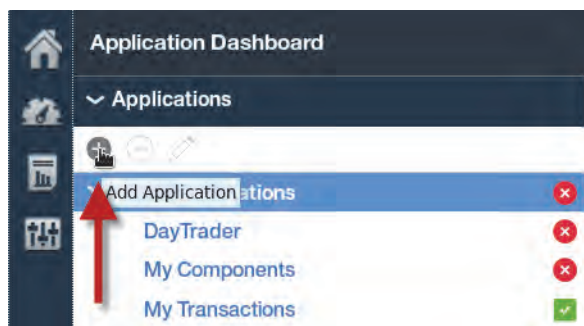
1. Move to the **Performance Management console** on the **apm** image.
2. If necessary, log in as **root** user with password **object00**.

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

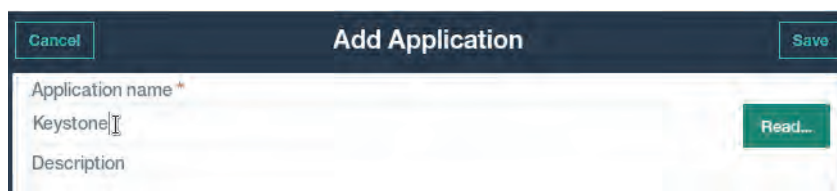


## Exercise 6 Creating the Keystone application in the Performance Management console

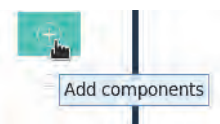
1. Click the plus (+) sign in the **Applications** widget.



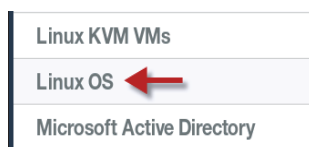
2. Type **Keystone** in the **Application name** field.



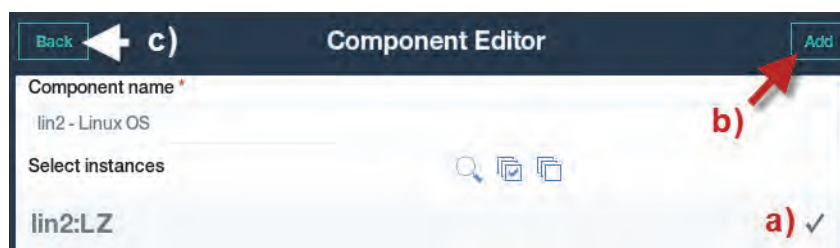
3. Click the plus (+) sign to add a component.



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



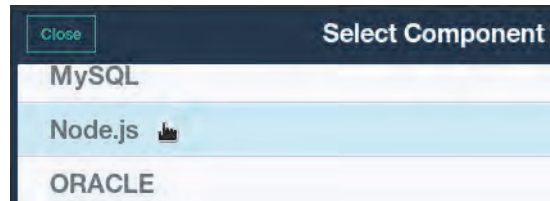
5. Add the single instance to the application.



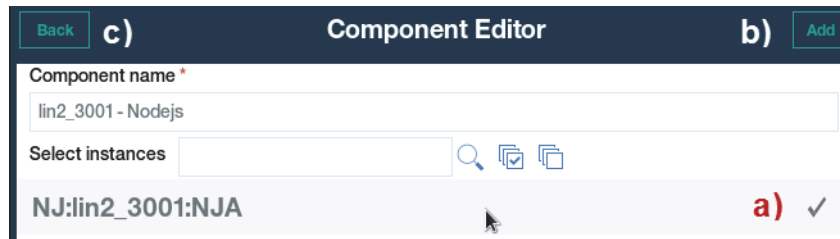
- a. Click the instance.
- b. Click **Add**.

## Exercise 6 Creating the Keystone application in the Performance Management console

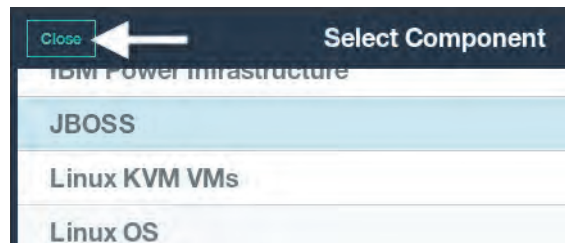
- c. Click **Back** to complete the operation.
6. Select **Node.js** in the list of available components.



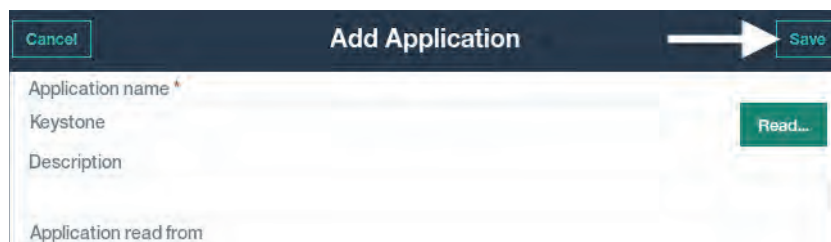
7. Add the single instance to the application.



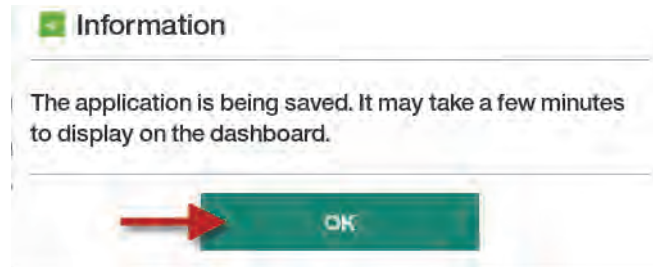
- a. Click the instance.
- b. Click **Add**.
- c. Click **Back** to complete the operation.
8. Click **Close**.



9. Click **Save**.

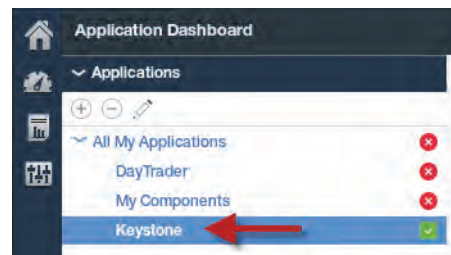


10. Click **OK** to complete the operation.

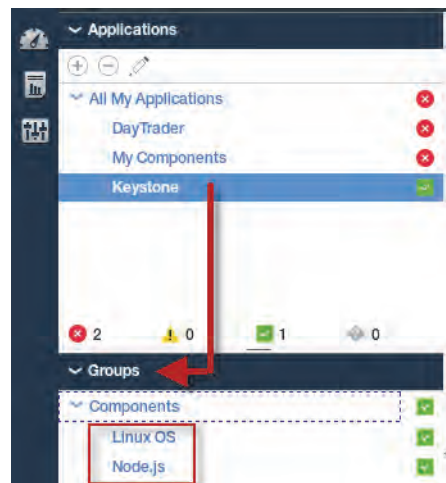


You are returned to the Application Dashboard and the **Keystone** application is displayed.

11. Confirm the creation of your application.



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



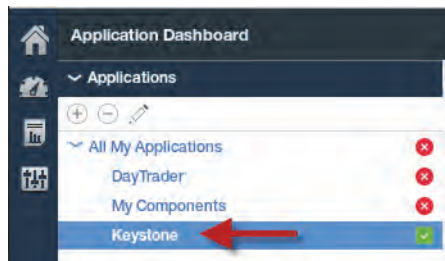
13. Verify that your components are listed in the **Current Components Status** widget.



**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.



- Click the Node.js instance widget.

lin2_3001 - Nodejs	
Application path	/downloads/node_app/key..
Request rate (RPM)	2
Average response time (ms)	⚠ 992
Slowest response time (ms)	No data available
Up time	9d 17h 52m 7s
CPU usage (%)	✅ 2.2
Memory usage (MB)	55
Application type	single

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







