Lab 3 Protecting your application environment with health policies

Health management is the ability of the system to take a policy-driven approach to monitoring the application server environment and taking action when certain predefined criteria are discovered. The health monitoring and management subsystem continuously monitors the operation of servers to detect functional degradation that is related to user application malfunctions. A health policy defines a set of conditions that are interpreted by WebSphere Application Server Intelligent Management as a degradation of server function. Health policies are a combination of a health condition to monitor, actions to take if the condition occurs, the deployment target to be monitored, and a reaction mode that defines whether the action occurs automatically or with operator intervention.

In this lab, you will deploy an application to a dynamic cluster and define a health policy to monitor the application for excessive memory usage. The health management policy will monitor a server instance for a condition when 70% of the heap has been exceeded for more than 1 minute. When this condition is breached, WebSphere Application Server Intelligent Management will start additional application resources, route new work to these new resources, and drain the troubled resource – while maintaining continuous application availability. You will use a servlet application that has been designed to continuously leak memory, slowly reducing the amount of free heap. You will monitor the heap usage visualizing using WebSphere Application Server Intelligent Management's charting capabilities.

1.1 Configure your environment for health management

In this section, you will configure your Java heap size, install an application, configure a health policy to monitor for excessive memory usage, and define a report with a chart to allow you to monitor memory usage in your environment.

1.1.1 Set the Java heap size

__1. Log into the administrative console for the deployment manager.

__a. Open a browser and point it to the Integrated Solutions Console https://labvm.ibm.com:9043/ibm/console

__b. Log in to the console with the username **admin** and a password of **admin**.

__2. Stop your dynamic cluster instances.

__a. Navigate to **Servers > Clusters > Dynamic clusters**. Notice that the **DynamicCluster01** is in **Automatic** mode.

__b. **Select** the box next to **DynamicCluster01 _0**, verify that the mode menu is set to **Manual**, then click the **Set Mode** button.

__c. Navigate to **Servers > Server Types > WebSphere Application Servers**. Select both boxes to all of your running cluster members, then press the Stop button.

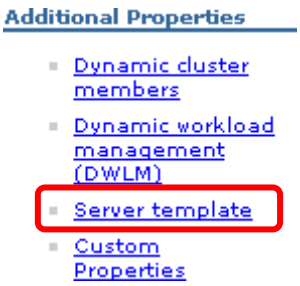
__d. On the Stop server verification page, select **OK**. On the feedback page, click **OK** again to return to the server list.

__e. After a moment, your servers should all be listed as stopped.

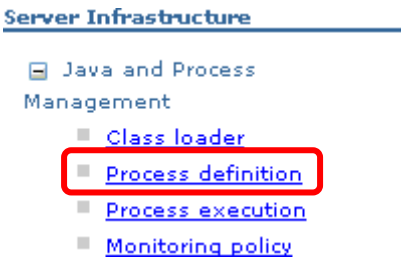
__3. Set the Java heap size for your dynamic cluster to 128 MB.

__a. Navigate in the left-hand frame to **Servers > Clusters > Dynamic Clusters**. Click on the link for the **DynamicCluster01** dynamic cluster.

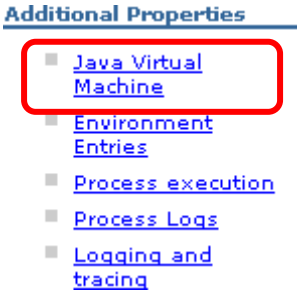
__b. On the right-hand of the screen under **Additional Properties** click on the **Server Template** link.



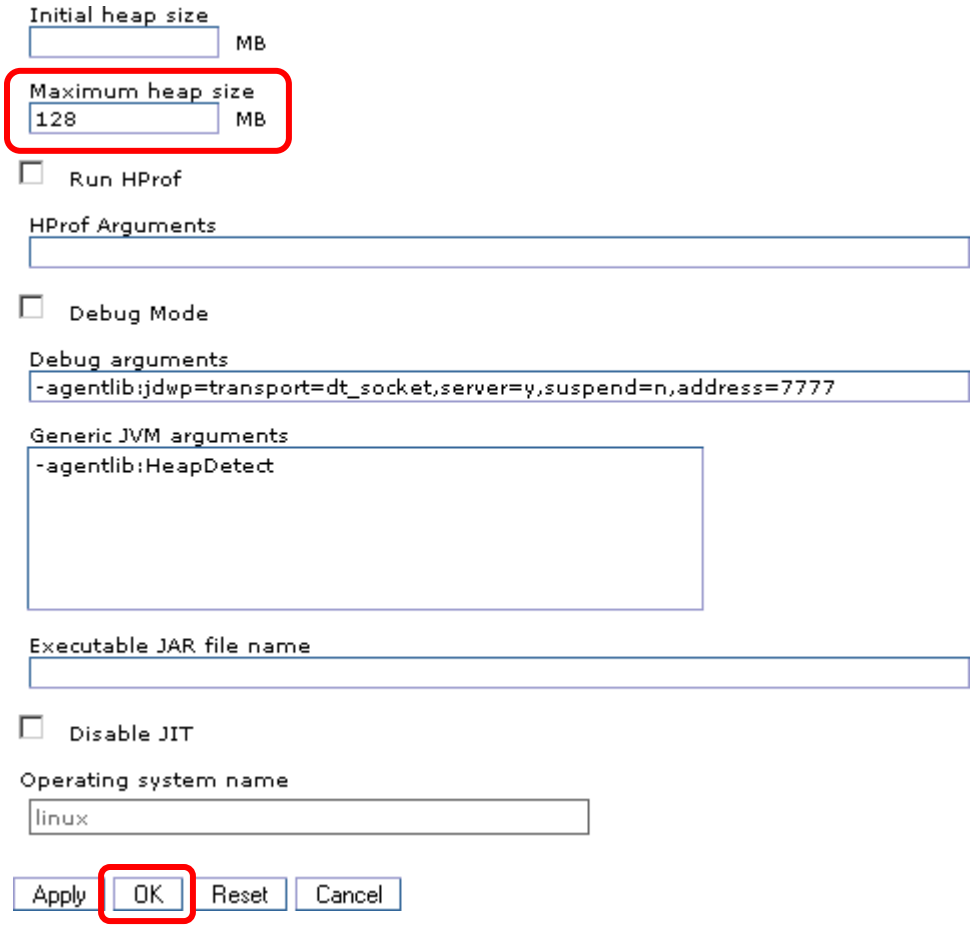
__c. On the Server Template page under **Server Infrastructure** expand the **Java and Process Management** tree click on the **Process Definition** link.



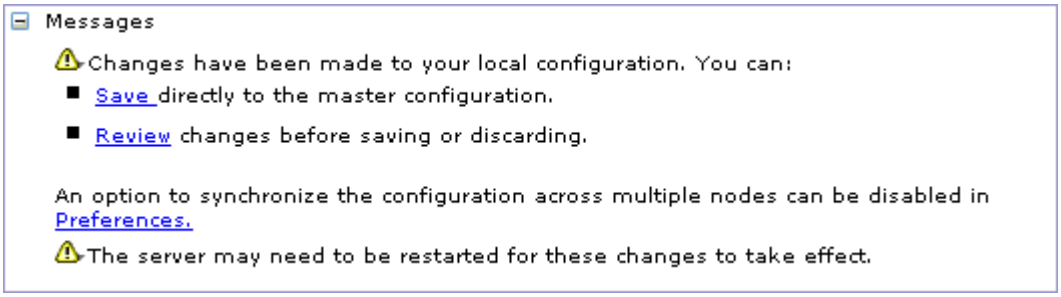
__d. On the Process Definition screen under **Additional Properties** click on the **Java Virtual Machine** link.



__e. On the Java Virtual Machine screen set the **Maximum heap size** to **128** and click the **OK** button.



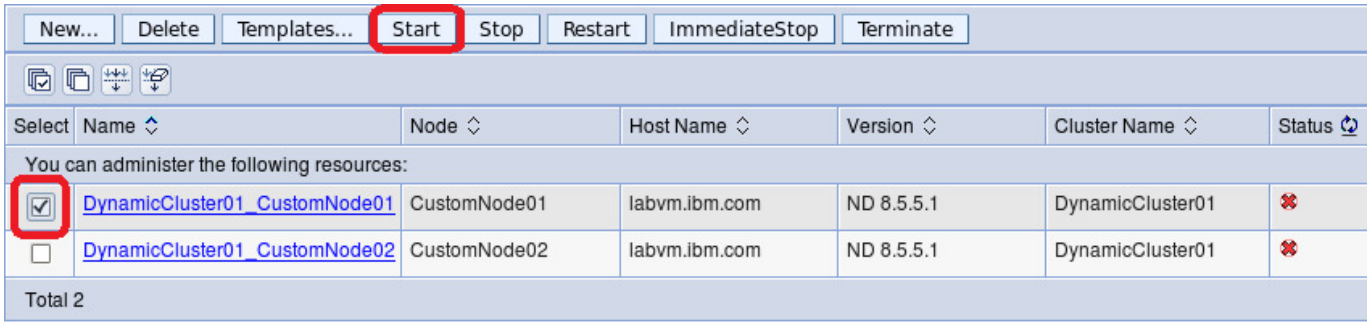
__f. Click the **Save** link at the top of the page to save the changes to the configuration repository.



__g. After the **Synchronize changes with nodes** processing completes, click **OK**.

__4. Restart your dynamic cluster server to capture the Java heap memory change.

__a. Go to the **Severs > Server Types > WebSphere applications servers** link. Select the first server in the list, then click the **Start** button.



__b. After a moment, you will see a message confirming that your server has started. Make a mental note that it is the first server in the list that is currently started – later in the lab, the second server will be running to help maintain your application availability when your application is leaking memory.

__c. Switch your dynamic cluster into **Supervised** mode, by navigating to **Servers > Clusters > Dynamic clusters**. Select the box next to **DynamicCluster01**, set the **Mode** dropdown to **Supervised**, then click the **Set Mode** button.

1.2 Install an application in your dynamic cluster

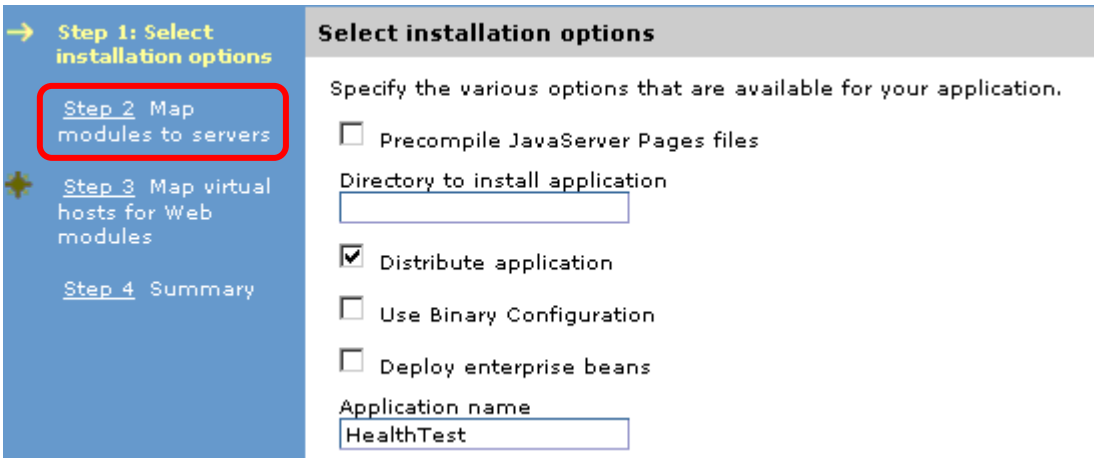
Now that you have configured your Java heap setting, you are ready to install your health policy test application.

__1. Go to the **Applications > New Application** link and click the **New Enterprise Application** link.

__2. Click the **Browse** button and navigate to **the /home/wasadmin/workshop/labs/HealthMgmt** folder and choose the **HealthTest.ear** file and click the **Open** button. Click the **Next** button to continue.

__3. On the **Enterprise Applications > Preparing for the application installation** screen, leave the **Fast Path** option selected and click the **Next** button.

__4. On the **Select installation options** screen click the **Step 2** link.

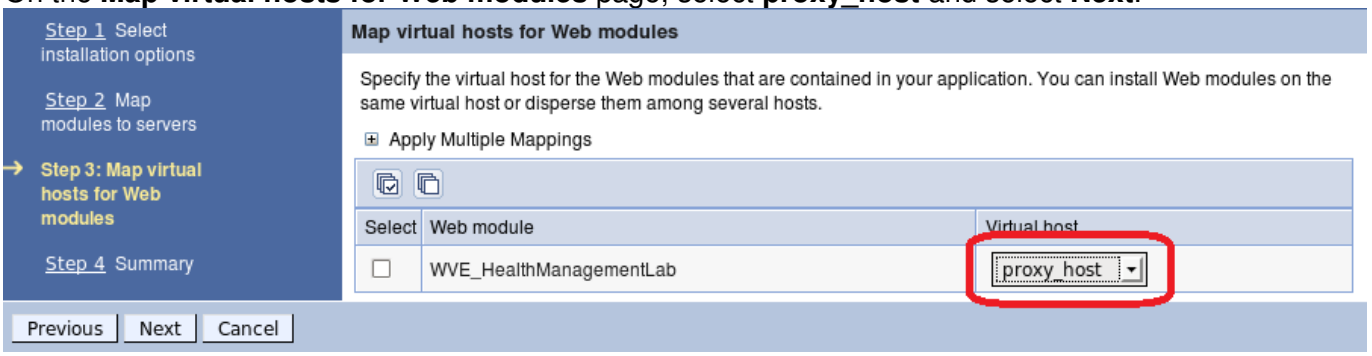


__5. On the **Map modules to server** screen, verify that the **WVE_HealthManagementLab** module is mapped to the **DynamicCluster01** cluster. If the **WVE_HealthManagementLab** module is not mapped to the **DynamicCluster01** dynamic cluster by default, then update it using these steps:

__a. Select the box in the **Select** column next to the **WVE_HealthManagementLab** module. Set the **Clusters and servers** menu to your **DynamicCluster01** cluster, then click the **Apply** button.

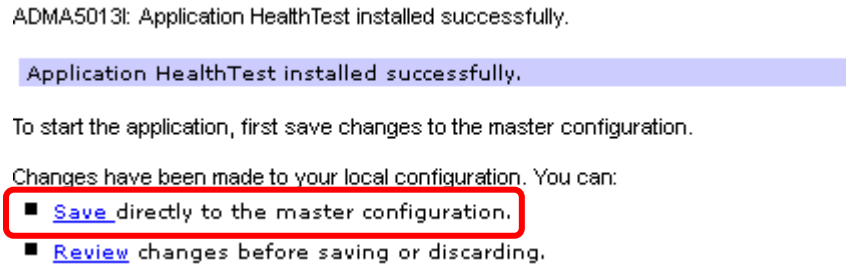
__6. Select **Next**.

__7. On the **Map virtual hosts for Web modules** page, select **proxy_host** and select **Next**.



__8. Scroll to the bottom of the **Summary** page and click the **Finish** button.

__9. Make sure the application installed successfully and click the **Save** link.



__10. After the **Synchronize changes with Nodes** processing completes, click the **OK** button. Now your application is installed!

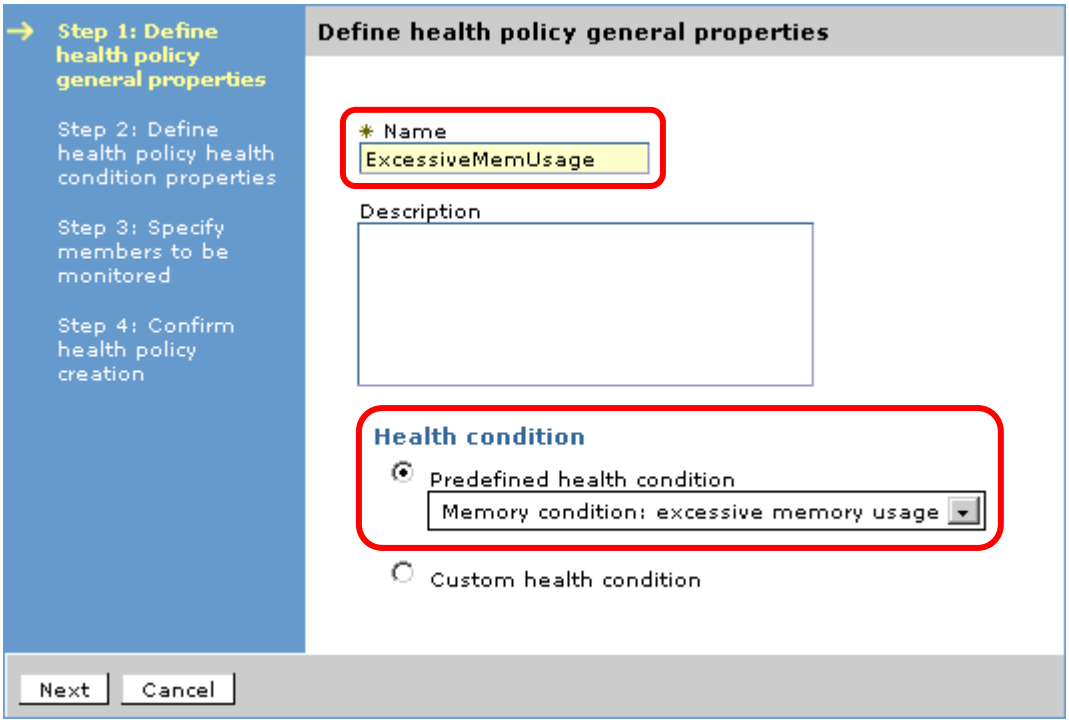
1.3 Create a health policy and configure reports

Next, you will create your health policy for excessive memory usage and configure a report to monitor the memory usage in your environment.

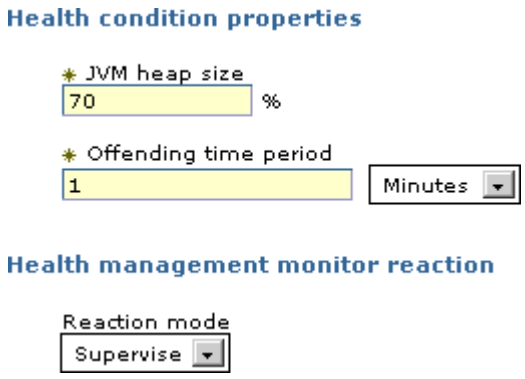
__1. Create a health policy.

__a. Go to **Operational policies > Health Policies** link. On the **Health Polices** screen click the **New** button.

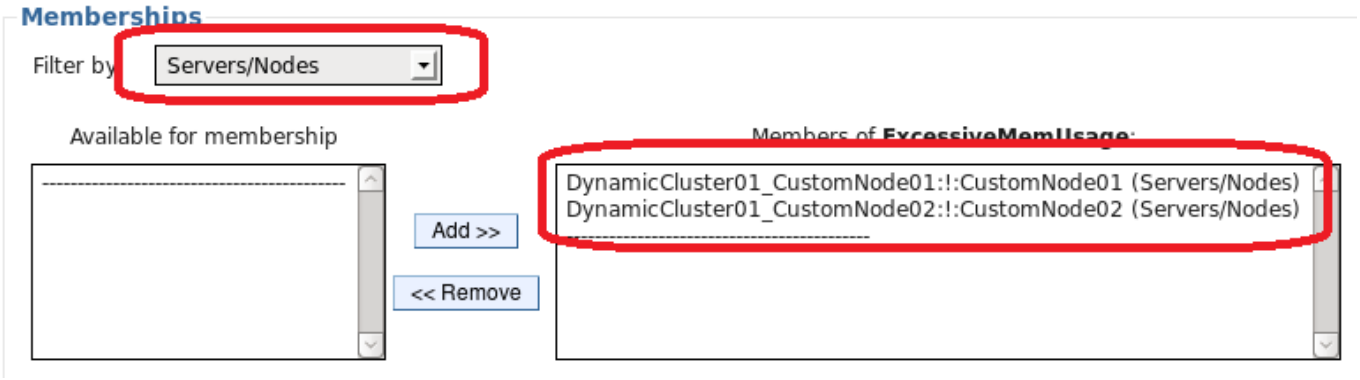
__b. On the **Step 1: Define health policy general properties** screen fill in **ExcessiveMemUsage** in the **Name** text box and chose **Memory condition: excessive memory usage** from the **Predefined health condition** drop down list. Click the **Next** button.



__c. On the **Step 2: Define health policy general properties** screen set the **JVM heap size** percentage to **70** and the **Offending time period** to **1** minute. Chose **Supervise** from the **Reaction mode** drop down list. Click the **Next** button.



__d. On the **Step 3: Specify members to be monitored** screen chose **Servers/Nodes** from the **Filter by** drop down list. Highlight both nodes under the **Available for membership** list and click the **Add >>** button to add the two nodes to the **Members of ExcessiveMemUsage** health policy list. Click the **Next** button



__e. Review your configuration and click the **Finish** button.

__f. Click the **Save** link and wait for the node synchronization to complete, then click the **OK** button. Now that your application is installed, you will create a chart to monitor memory usage.

__2. Create a report to chart and track the Java heap memory utilization of your dynamic cluster members.

__a. Go to the **Runtime Operations** > **Reports** link.

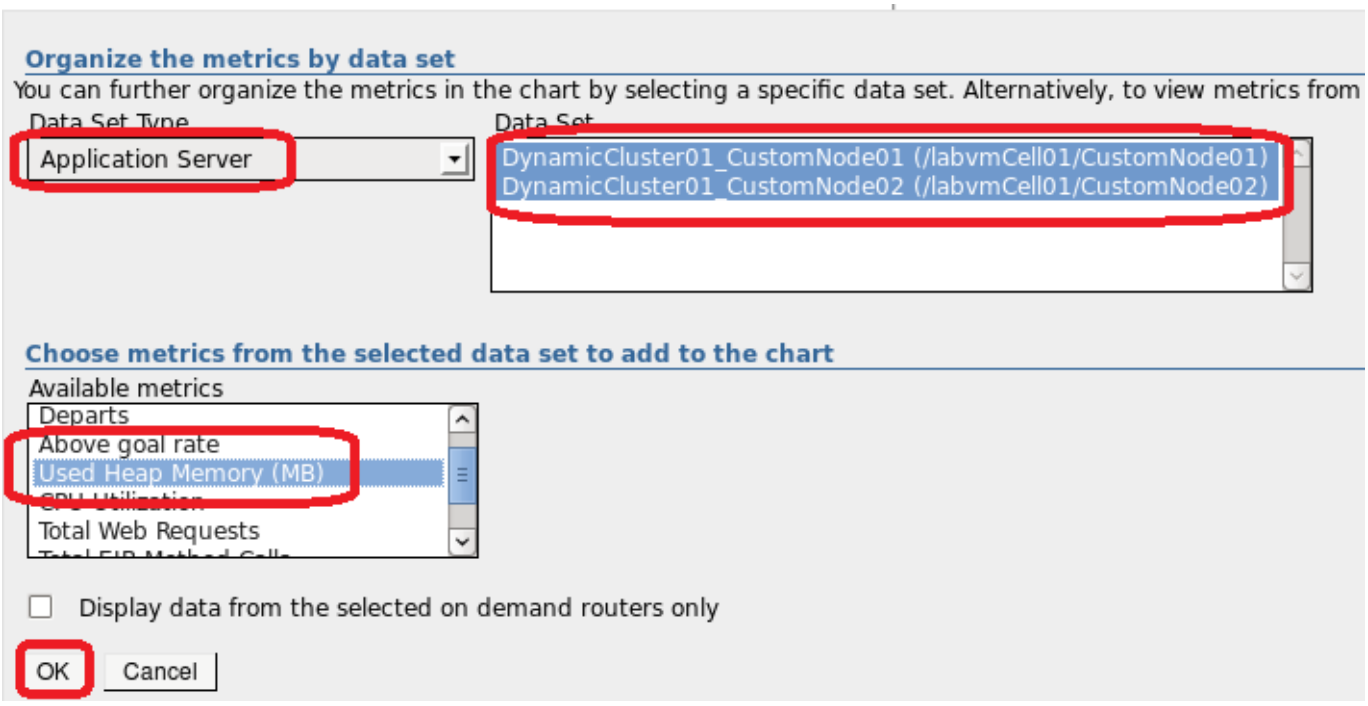
__b. On the chart page, scroll down and click on the **Add Data…** button. Be sure to scroll all the way down – it is near the bottom of the page.

__c. In the **Organize the metrics by data set** panel, use these values:

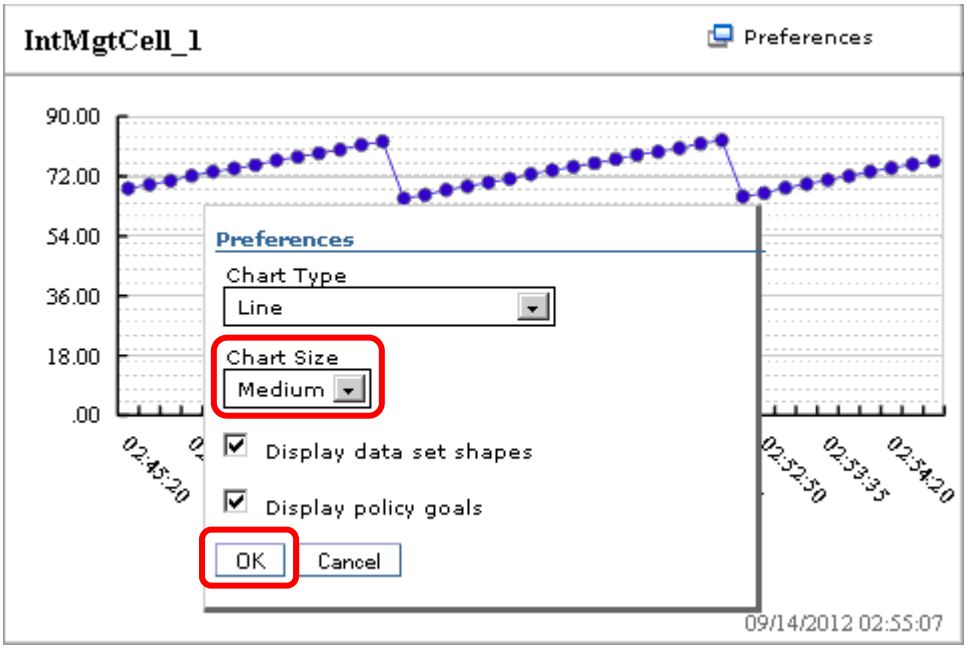
__i. From **Data Set Type**, chose **Application Server**.

__ii. Select both nodes in the **Data Set** list (use **Shift+Click** to select multiple items in the list).

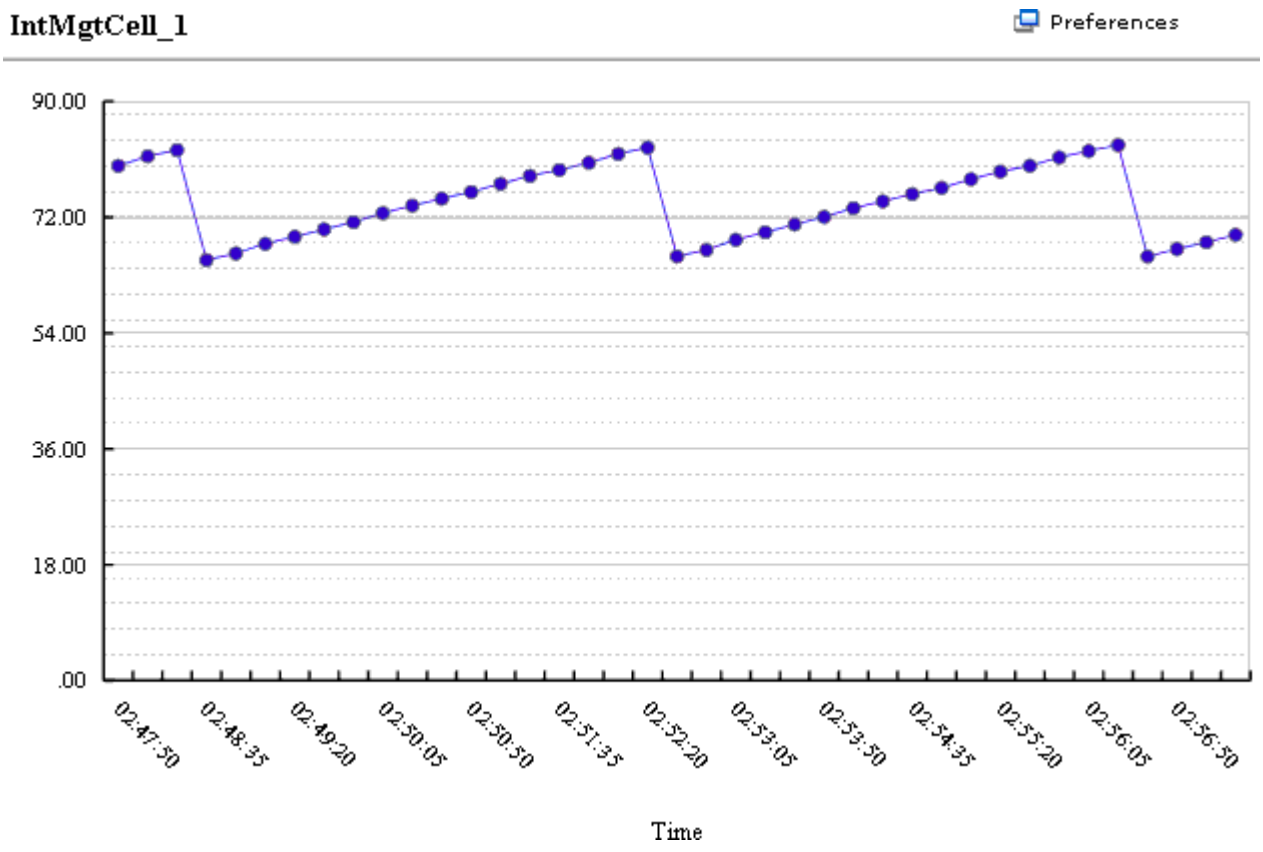
__iii. Scroll through the **Available metrics** list and highlight **Used Heap Memory (MB)** from the list. Click the **OK** button.



__d. Click the **Preferences** link on the chart and chose **Medium** for the **Chart Size** and click the **OK** button.



__e. You should see a chart that shows one server running with a nice saw tooth-shaped Java heap memory utilization curve.



__f. In case you have to log out of the console or start a new session, it is a good idea to save your chart group to make it easy to open later. Near the bottom of the display, locate the **Save current group of chart tabs configuration as a chart group** field and type **Memory Usage Charts** into the field. Click the **Save** button. Choose to **Save** your changed to the master configuration, then choose **OK** after synchronization completes.

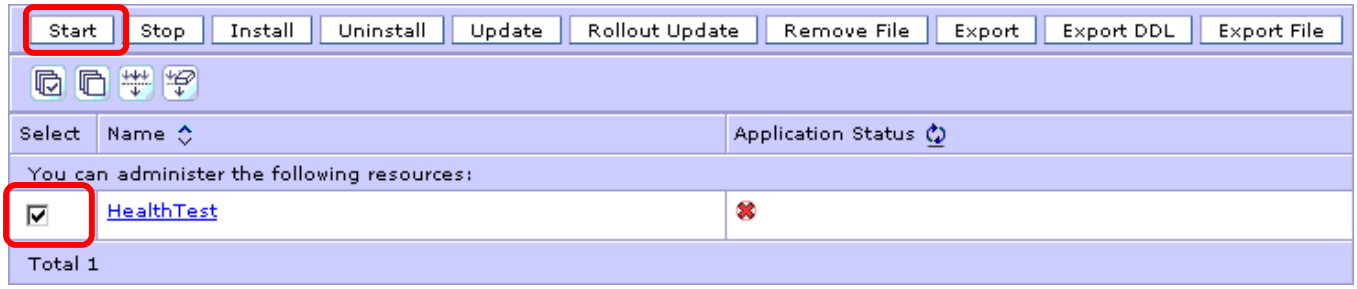
__g. At this point, you have a dynamic cluster with an application installed, a health policy configured for excessive memory usage, and a graph to allow you to monitor memory usage in your environment. You are ready to use your application to start leaking memory so that you can see your health policy in action.

1.4 Use your health policy to protect your environment

__1. Access your application and use it to leak some memory.

__a. Before trying to access the application, you need to start it. Navigate to **Applications > All Applications** link, select the box next to the **HealthTest** application, then click the **Start** button.

__i. If you do not see the application in the list, you can manually synchronize the nodes in your environment. Go to **System administration > Save changes to master repository**. Verify that the **Synchronize changes with Nodes** box is selected, then click the **Save** button. After synchronization completes, click **OK**. Now navigate back to the application page and you should see your application and be able to start it.



__b. After a moment, you will see a message that the application started successfully.

__c. Open a new browser tab and type in the following URL:

http://<ODR_IP>/VE-Health/HealthTestApp?amount=1000000&sleep=250&leakp=100

The ‘amount’ parameter tells the application how much memory to leak and the ‘sleep’ parameter tells the application how to long to sleep on the request. The ‘leakp’ parameter defines the probability of a memory leak. You will send requests to gradually leak memory at the rate of about one million bytes at a time. On each request, you will get a response telling us how much memory has been leaked thus far.

__d. Scroll down in the application landing page that loaded, until you find the **Result** section. You will see a message indicating the leak size of your last application request.

Result

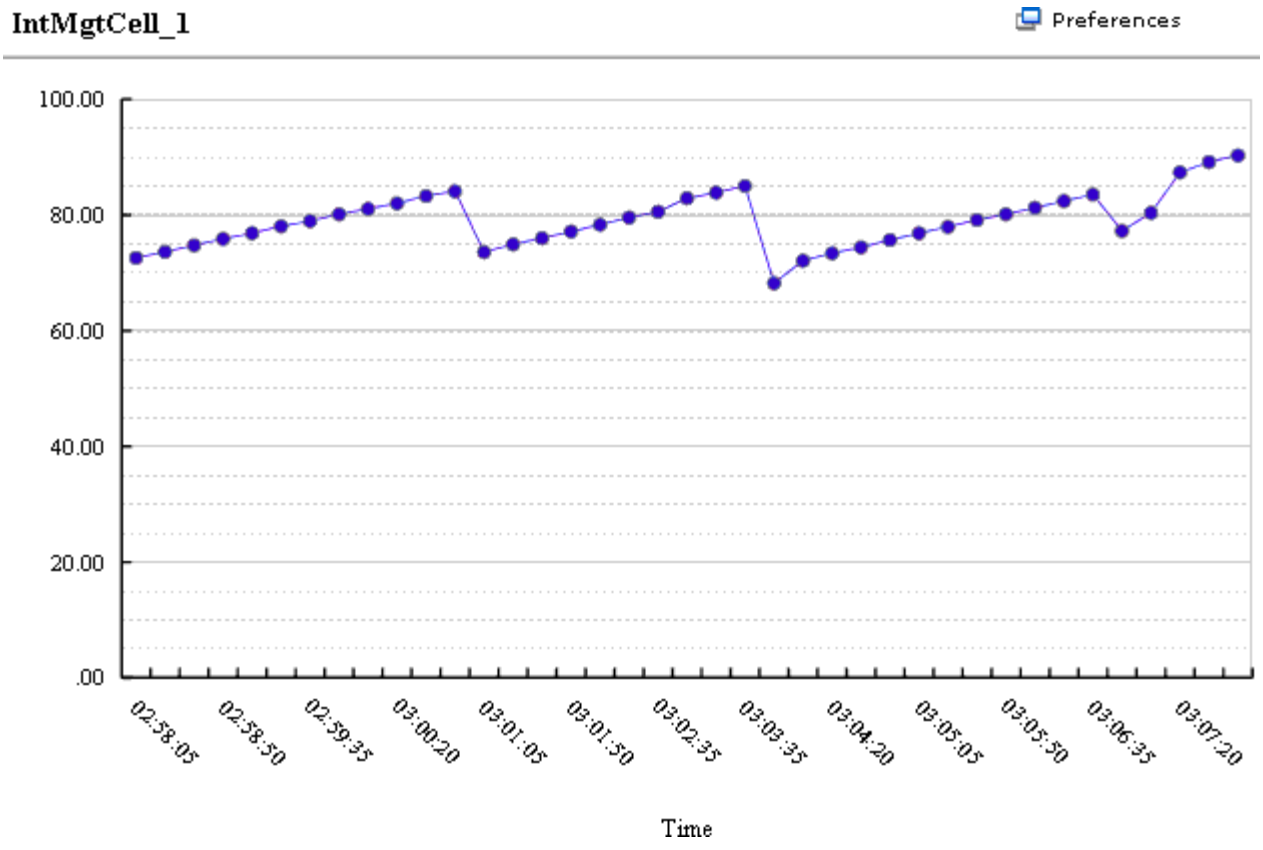
true

Servlet run time = 1,168 ms

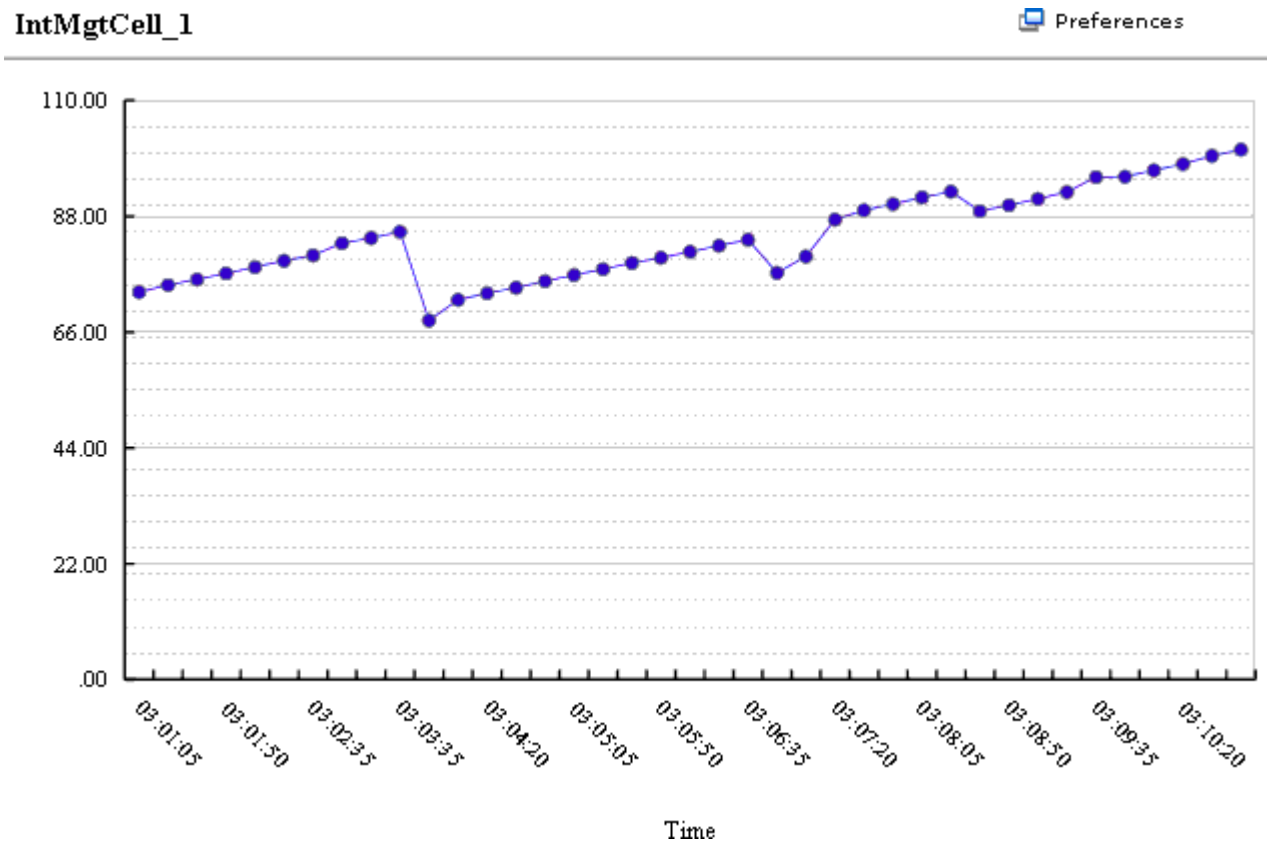
Leak size is 1,000,000 bytes, total 1 leaks, 1,000,000 bytes

__e. Refresh the browser 6 times to “leak” some more memory with the application.

__f. In the administrative console tab, navigate back to **Runtime Operations > Reports**, to view your memory usage report. You should see a growth in the heap utilization in the chart.



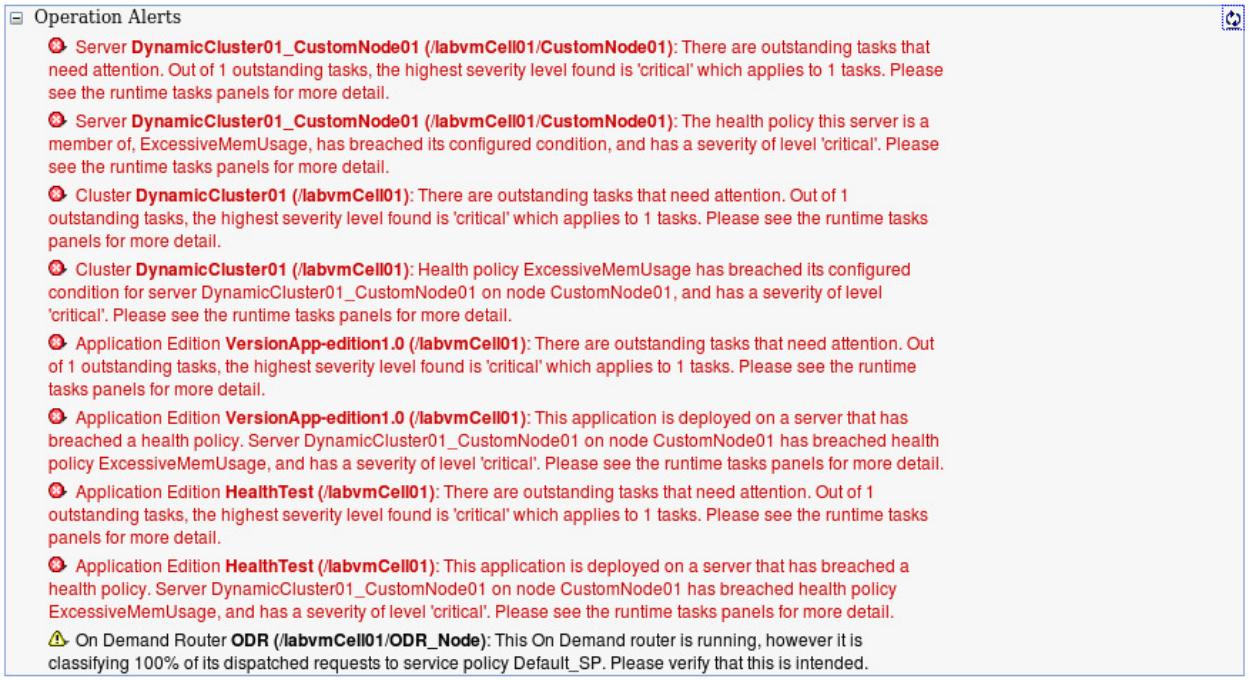
__g. Refresh the browser 6 more times to leak a little more memory. The chart should show an overall increase in heap utilization.

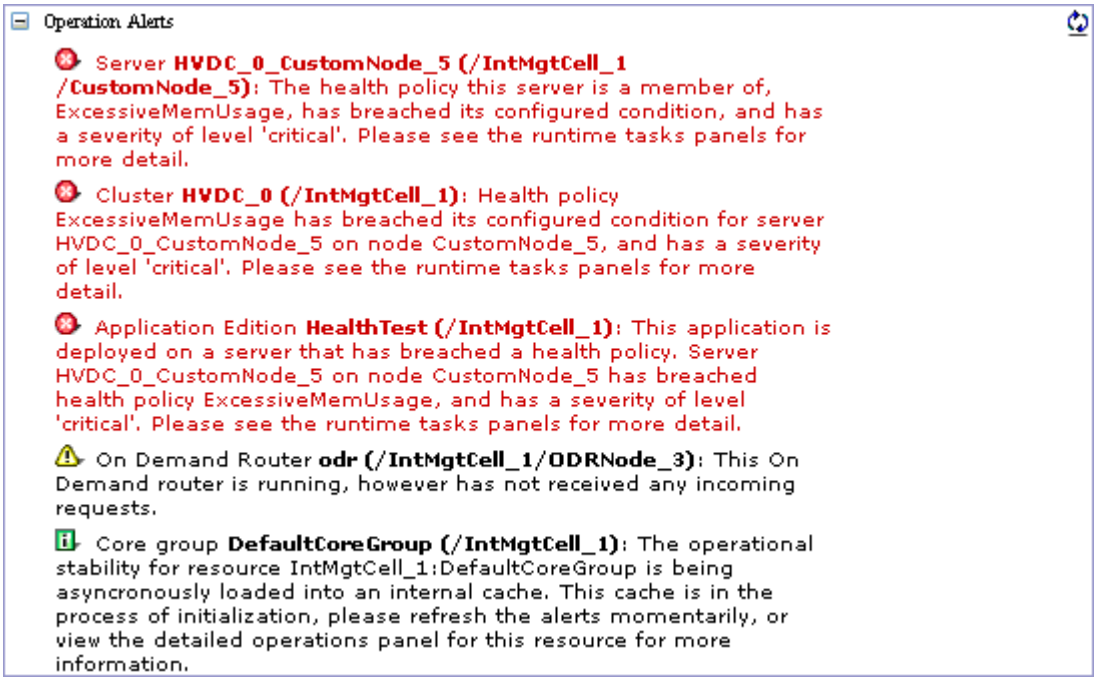


__2. Watch your health management policy take automatic corrective action.

__a. Your memory usage health policy is configured to trigger at 70% of heap usage – which in your case is 70% of 128MB, or approximately 90MB of memory. So, take a look at your graph and verify that it really is hovering above 90MB. There will be some garbage collection cycles, and overall memory usage might go down slightly, but keep an eye on things to make sure that the memory usage is high enough to kick off the policy.

__b. After 5 to 10 minutes you should that the health management policy has started the other server in your dynamic cluster. You will also see red operational alerts in the **Operations Alert** section at the top of the chart indicating that the application server has breached its health management policy.

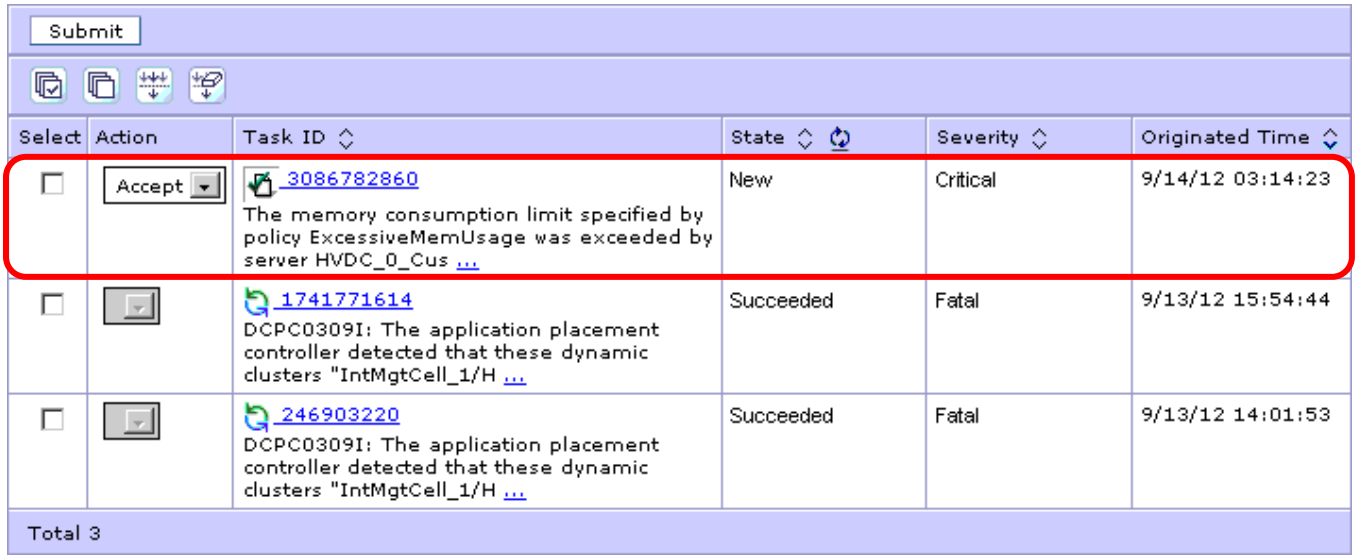




__3. Kick off the runtime task to take corrective action.

__a. Since your cluster and policy are configured in **Supervised** mode, you need to go submit the waiting runtime task in order for it to take corrective action.

__b. Navigate to **System Administration > Task Management > Runtime Tasks**, and locate the **New** task at the top of the table.



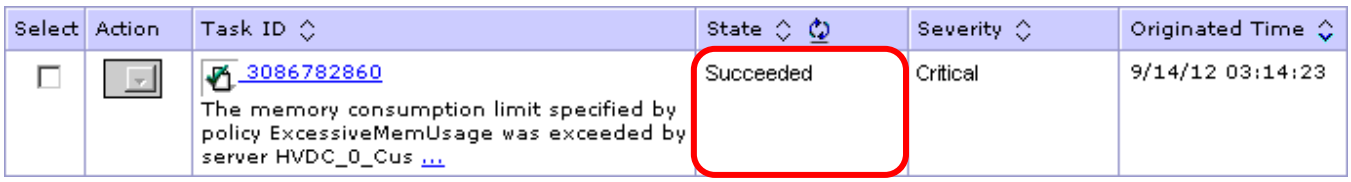
__c. Click the number of the **Task ID** for your new task to open its details.

__d. Look near the bottom of your runtime task details to find the **Action plan to resolve the situation**. Your action plan will consist of a single step – a server restart action to get rid of the leaky server.

Step 1 : Execute the command AdminTask.healthRestartAction(["-clusterName","null","-cellName","labvmCell01","-nodeName","CustomNode01","-serverName","DynamicCluster01_CustomNode01"]) .

__e. To kick off the corrective action, select the Submit button near the top of the display. Navigate back to the **Runtime Tasks** list (use the context menu at the top of the panel, or go to **System Administration > Task Management > Runtime Tasks**). The **State** of your task should be shown as **In progress**.

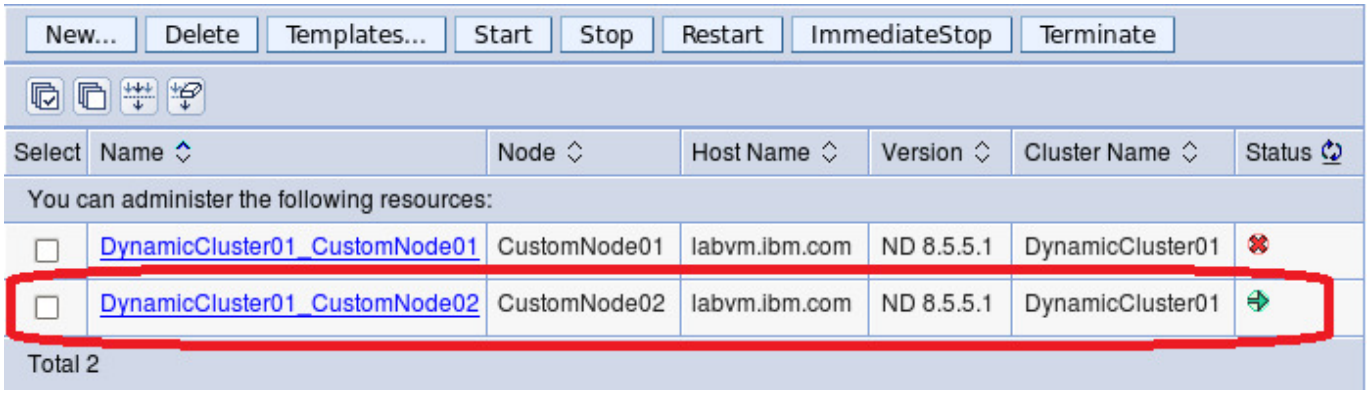
__f. Refresh the panel periodically. After a few moments, the **State** of your task will change to **Succeeded**.



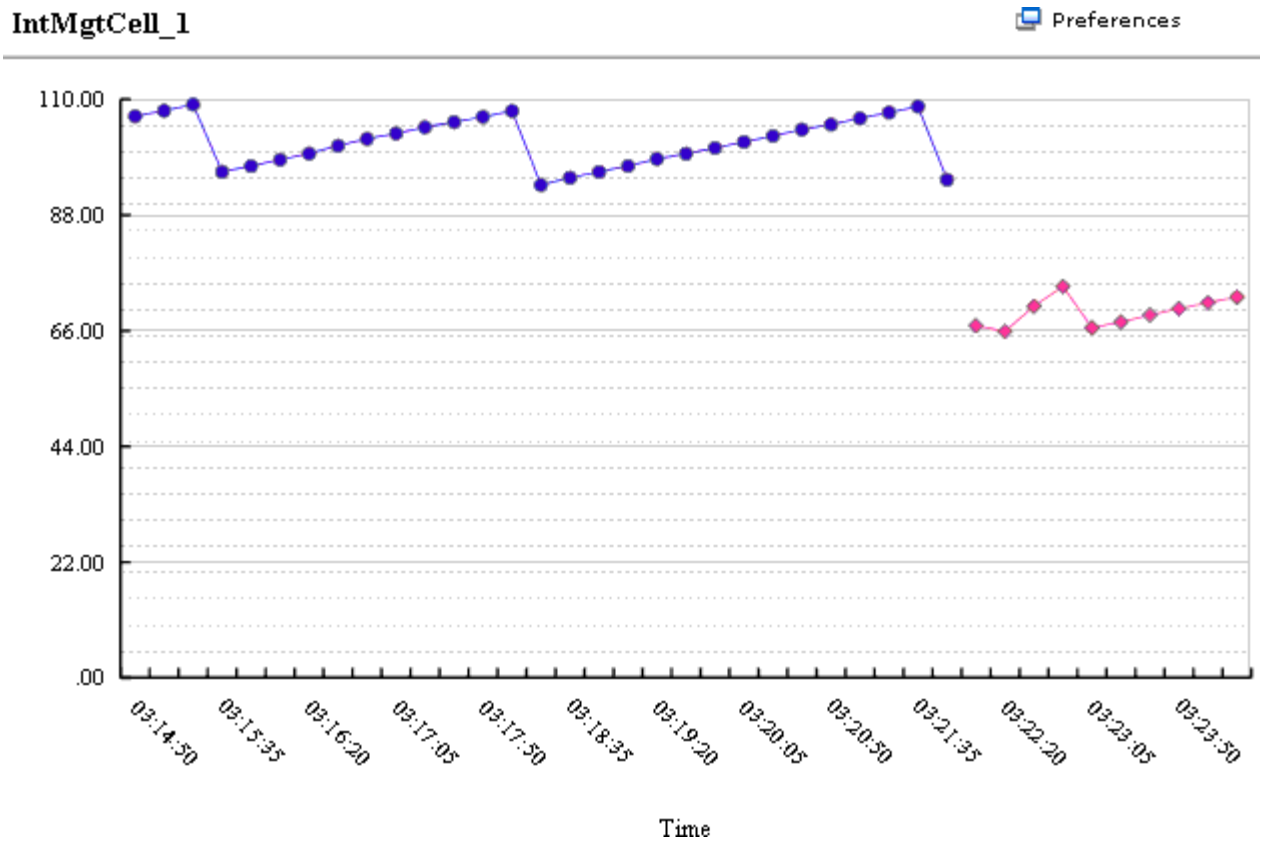
__g. Note that you had to take action to perform the health action because you had your environment configured in **Supervised** mode. If you configure your environment in **Automatic** mode instead, it will take corrective action without any operator intervention!

__4. Verify that you have a server running and that your application is active.

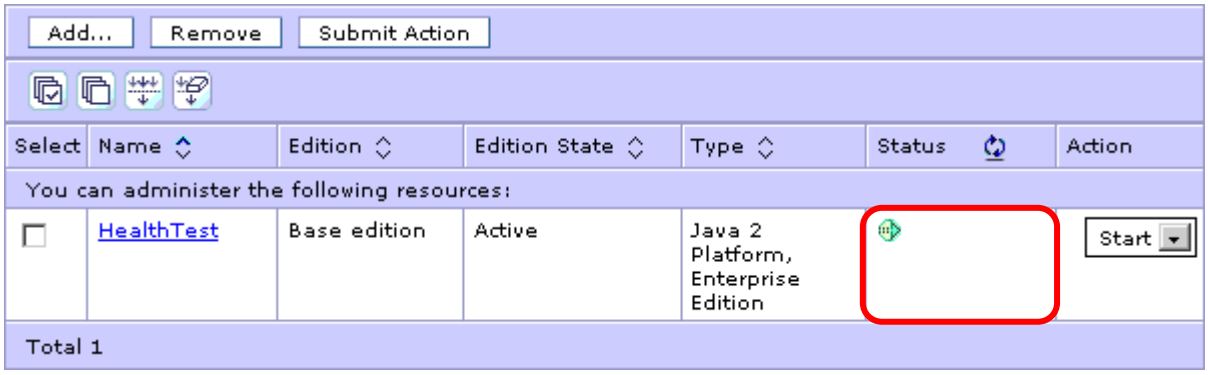
__a. Navigate back to **Servers > Server Types > WebSphere Application Servers** and note that you still have one application server running. It should be the second server in the list. In this case, the running server is the other cluster member than that server that was running originally. The second server was started up to provide continuous availability to your application, and the sick server with the leaking memory is no longer running.



__b. Look at your report under **Runtime Operations > Reports**. You have a new line in the graph with a different color, showing the new server getting started and starting to consume memory.

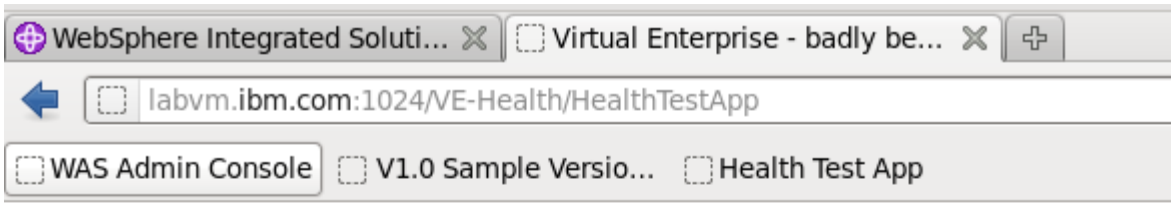


__c. Next, check that your application is really running by looking under **Applications > All Applications**. Your application should be in **Started** or **Partial Start** state, as shown.



__d. Finally, access your running application. You will still access the application through the on demand router. In your browser, use a URL of this form:

http://labvm.ibm.com:1024/VE-Health/HealthTestApp



HealthTestApp Request/Result

Based on CpuAndSleepBound

Parameters

deterministic:	true	(Default)
countMean:	1 ms	(Default)
countMax:	100,000	(Default)
sleepInterval:	1	(Default)
yieldInterval:	1,000	(Default)
sleepLength:	1,000 ms	(Default)
debConc:	true	(Default)
zk:	true	(Default)

Failure parameters

__e. Success! You are still able to access your application, after the Intelligent Management system took corrective action to recover from a memory leak.

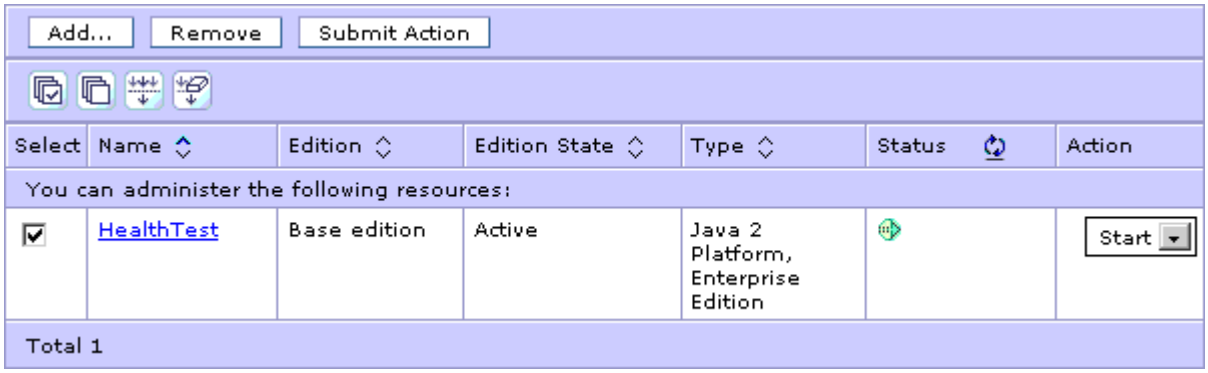
1.5 Clean up

Now that you are done with the lab scenario, you can clean up the environment.

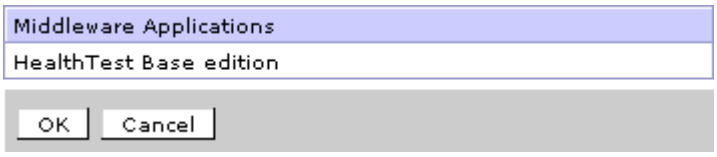
__1. Delete the **HealthTest** application.

__a. Navigate to **Applications > All applications**.

__b. Select the box next to the **HealthTest** application, then click the **Remove** button.



__c. Click **OK** to remove the application.

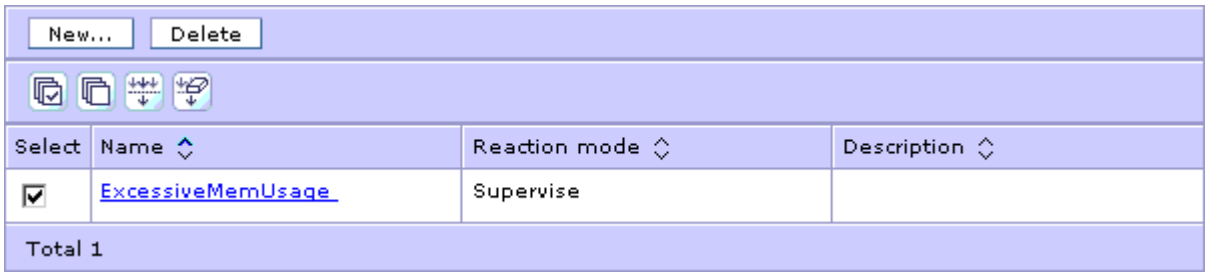


__d. Select **Save** to save your changes to the configuration repository. After synchronization completes, click the **OK** button.

__2. Delete your health policy.

__a. Navigate to **Operational policies > Health Policies**.

__b. Select the box next to **ExcessiveMemUsage**, then choose **Delete**.



__c. Select **Save** to save your changes to the configuration repository. After synchronization completes, click the **OK** button.

__d. Clean up is now complete.

1.6 Summary

In this lab, you learned how to use a health policy to automatically protect your server environment that was leaking memory. You used a simple application that was designed to quickly leak some memory, with a health policy configured to automatically recycle the sick server. You saw how to create a simple report to track memory utilization, and then watched your cluster automatically recover from a memory leak, while maintaining continuous application availability.