Course Exercises Guide

Administering WebSphere Application Server Liberty Profile

Course code WA190 / ZX190 ERC 1.0

IBM Training

November 2016 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered 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.

Contents

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

Bluemix®	DB™	developerWorks®
HACMP™	MVS™	Notes®
Passport Advantage®	Redbooks®	Tivoli®
WebSphere®	z/OS®	

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

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

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

Social® is a trademark or registered trademark of TWC Product and Technology, LLC, an IBM Company.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

This course includes the following exercises:

- Managing Liberty collectives with the Admin Center
- WebSphere Liberty administration by using Jython scripts
- Dynamic Routing
- Auto-scaling
- Using the IBM HTTP Server with SSL to a Liberty server

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections, which should always be completed. It might be necessary to complete these sections before you can start later exercises. If you have sufficient time and want an extra challenge, some exercises might also include optional sections that you can complete.

Important

Online course material updates might exist for this course. To check for updates, see the Instructor wiki at http://ibm.biz/CloudEduCourses.

Exercise 1. Managing Liberty collectives with the Admin Center

Estimated time

01:15

Overview

In this exercise, you learn how to configure a collective controller and use the Admin Center to manage Liberty servers from the collective controller. You deploy a cluster of packaged servers, view the deployment environment, and view basic performance metrics all by using the Admin Center.

Objectives

After completing this exercise, you should be able to:

- Create a collective controller
- Navigate in the Admin Center
- Deploy a cluster of packaged Liberty servers
- View the deployment environment
- View basic performance metrics

Introduction

IBM WebSphere Liberty is a lightweight, highly composable, fast to start, dynamic application server runtime environment. Liberty provides a dynamic, flexible run time for Java applications, by providing the complete Java EE 7 platform and a subset of the full WebSphere Application Server API.

Liberty is ideal for use in both development and production environments. Liberty is a good option for developers who are building web applications that do not require the full Java EE environment of traditional enterprise application server profiles. Each runtime instance can be customized to match the needs of the application. In production environments, enterprise qualities of service, such as security and monitoring, are enabled as required.

Requirements

To complete this exercise, you need the WebSphere Liberty binary files and Liberty installed.

Exercise instructions

Part 1: General exercise information

This section provides general information about the exercises in this course. Review this section before starting the exercises.

User accounts

Type	User ID	Password
Operating system	localuser	passw0rd
Liberty	admin	passw0rd

An Ubuntu user ID was created for you. You use this ID to log on to the image.

- User ID: `localuser`
- Password: `passw0rd` (replace the o with a zero 0)

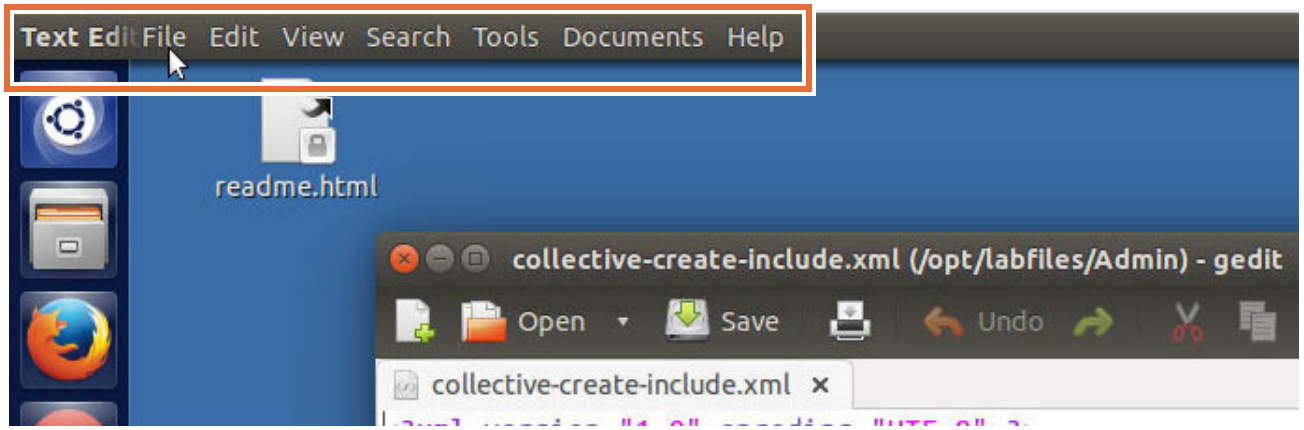
Information

The supplied course image is Ubuntu 14.04 LTS. The desktop uses Unity, which is different than the common Gnome desktop. Some hints on using Unity are at:

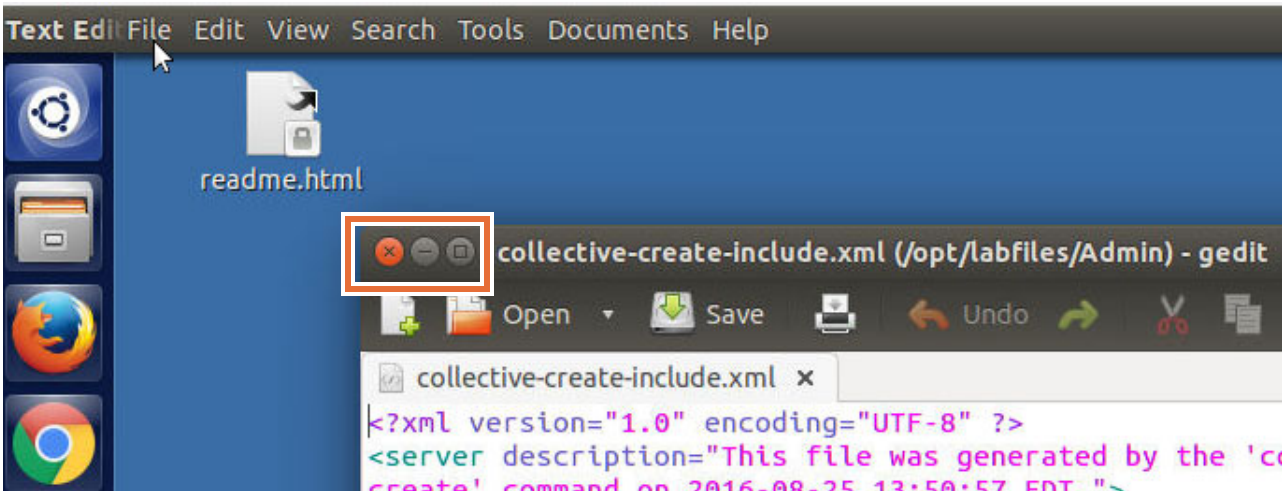
http://www.howtogeek.com/113330/how-to-master-ubuntus-unity-desktop-8-things-you-need-to-know/

Information

The desktop for Ubuntu is the Unity desktop. Unity uses a global menu. Which means application menus are not located in the window for the application. They are on the top pane. When a window is the active window, that window does not have any menu items, but the application type is displayed in the black bar that spans the top of the desktop. You cannot see the menu for the application until you hover your mouse over the top pane. When you hover your move over the black bar, the menu items for the active window are displayed.



In the corner of the application, you have the close, minimize, and full screen options.



When you maximize the application window, the close, minimize, and full screen options also appear in the top pane.

Course labfiles

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, PI files, and others. The course lab files can be found in the following directory:

- `/opt/labfiles` for the Linux operating system

The exercises point you to the lab files as you need them.

Stop

Course updates and corrections

A course corrections document might be available for this course.

If you are taking the class with an instructor, the instructor can provide this document to you.

If your are taking the course in a self-paced environment, the course corrections document is provided with the other manuals.

To check whether a course corrections document exists for this course:

1. Go to the following URL: http://ibm.biz/CloudEduCourses.
2. On the web page, locate, and click the **Course Information** category.
3. Find your course in the list and click the link.
4. Click the **Attachments** tab to see whether a course corrections document exists with updated instructions.
5. To save the file to your computer, click the document link and follow the prompts.

Part 2: Working with the Liberty profile settings

Information

You can install Liberty by using the Installation Manager, by extracting downloaded archive files, or by using the WebSphere Application Server Developer Tools for Eclipse. Each installation method offers different benefits.

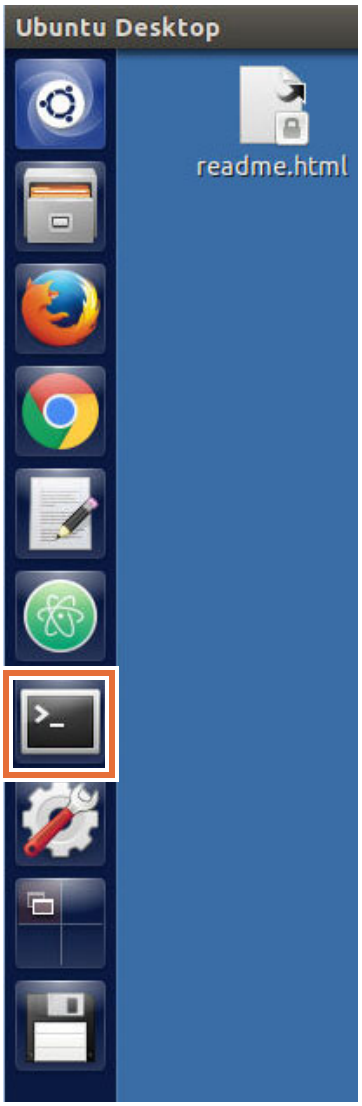
On the course image, Liberty is already installed. The method that is used to install Liberty was by extracting downloaded archive files, which is a JAR file. The following command was used to install Liberty:

```
java -jar /opt/WLP_16.0.0.2/Liberty/wlp-nd-all-16.0.0.2.jar
```

To install all features that apply your Liberty edition, you can install a feature bundle addon. In this part of the exercise, you install the ndMemberBundle for collective member servers and ndControllerBundle for collective controllers.

__ 1. Examine the environment.

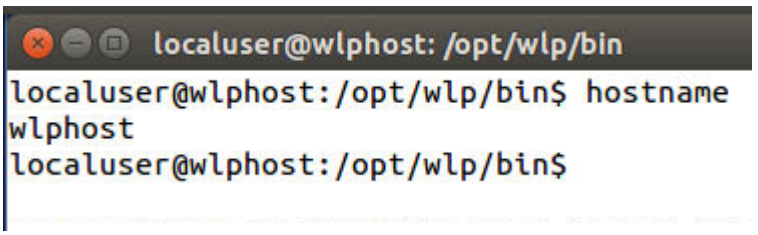
__ a. Open a terminal window. Click the **Terminal** icon in the toolbar, which is the left area of the image.



__ b. Go to the `/opt/wlp/bin` directory.

__ c. In the terminal window, determine the host name by entering the following command:

```
hostname
```

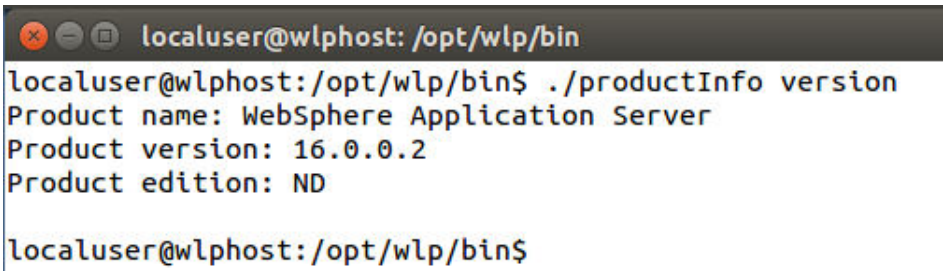


Information

WebSphere Application Server offers a continuous delivery model to deliver new features and functions to Liberty. The continuous delivery model provides new optionally installable features and functions, which can be added to an existing Liberty installation at the current service level with no requirement for a version upgrade or migration. The continuous delivery model allows IBM to deliver features at regular intervals so you do not have to wait for these new technologies to be released at the next major release.

__ d. Determine the version of Liberty that is on the course image. To get the version information, enter the following command:

```
./productInfo version
```



You can see that the version of Liberty is 16.0.0.2 that is installed on the course image.

Information

The Liberty Repository provides an online mechanism to deliver Liberty and more content, enabling a single point of access for various asset types. The Liberty Repository provides early access to supported new content, including new product capabilities, when they are delivered, rather than waiting for a new release.

In addition to features, the repository also includes artifacts, such as administration scripts, samples, configuration snippets, and artifacts that integrate open source projects more quickly and effectively.

A few ways that you can access the online Liberty Repository:

- From the Downloads page on the WASdev.net website
- From within the developer tools
- By using the Installation Manager and command line utilities such as the `installUtility` command

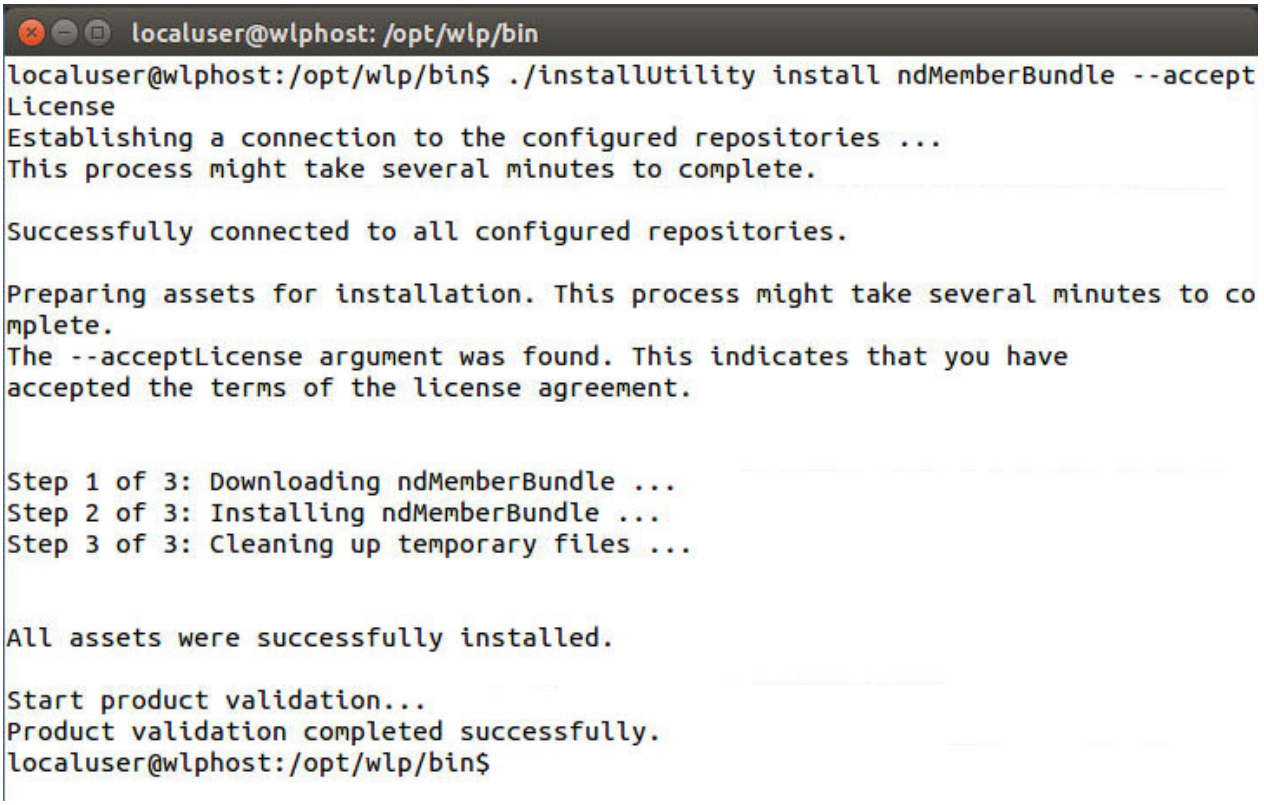
In the next step, you use the `installUtility` command to install all features for WebSphere Application Server Network Deployment Liberty. You install the ndControllerBundle addon on a server that manages Liberty collectives and the ndMemberBundle addon on a server that is clustered and auto-scaled in a Liberty collective.

Liberty is installed with the Network Deployment edition, which includes the ndControllerBundle and ndMemberBundle. However, you use the `installUtility` command to install the features to see how to use the command for downloading and installing bundles.

__ 2. Install the administration bundles.

__ a. Install the ndMemberBundle by entering the following command:

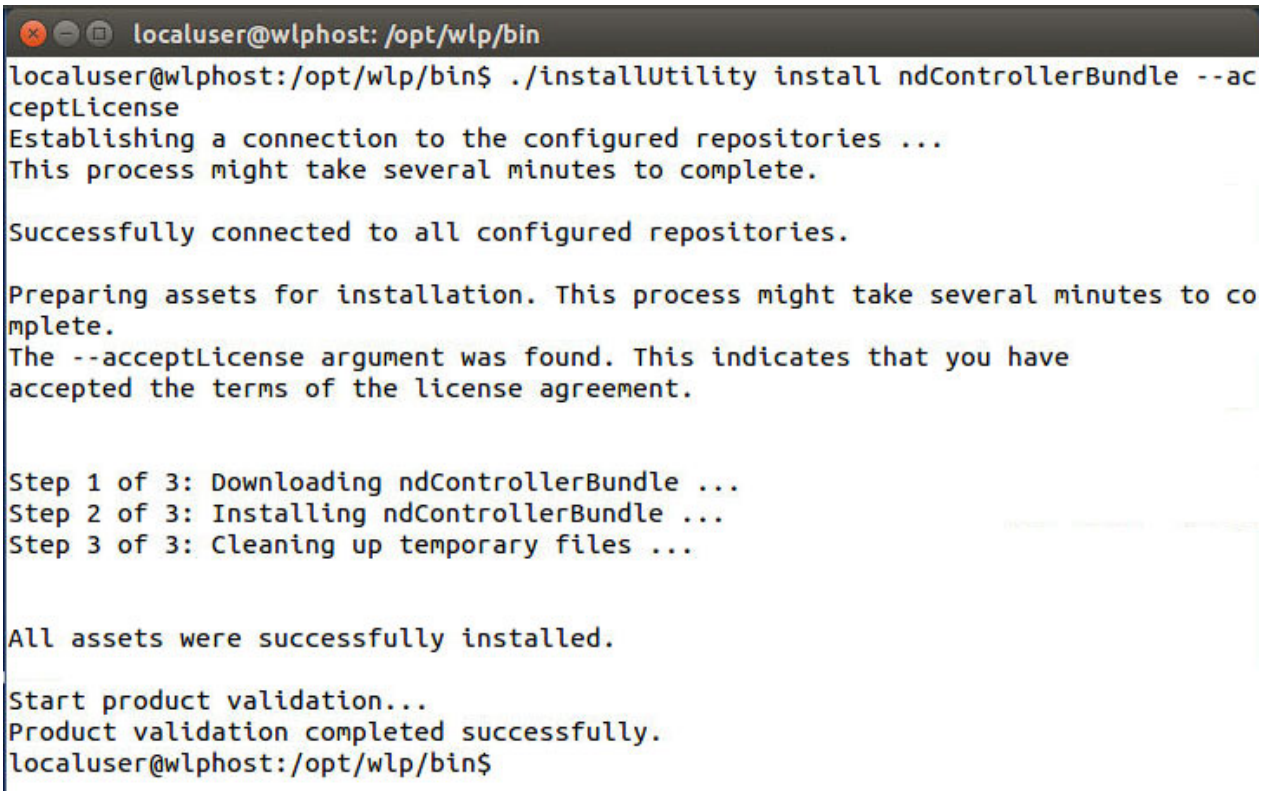
```
./installUtility install ndMemberBundle --acceptLicense
```



Verify that all assets are successfully installed.

__ b. Next, install the ndControllerBundle by entering the following command:

```
./installUtility install ndControllerBundle --acceptLicense
```



Verify that all assets are successfully installed.

__ 3. Encode the password and review the settings.

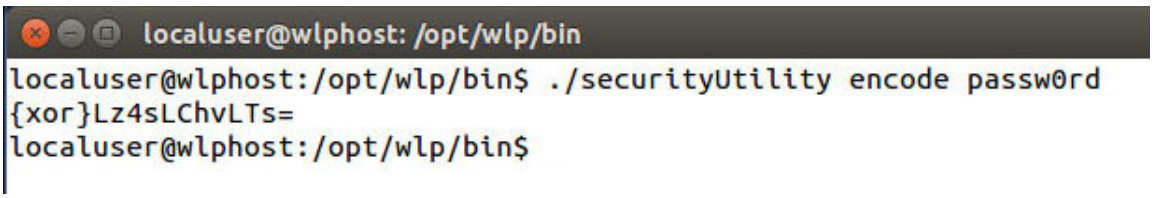
Information

Liberty supports Advanced Encryption Standard (AES) encryption for passwords that are stored in the `server.xml` file. The Liberty profile also supports password encoding and password hashing. Password encoding uses a simple algorithm to obfuscate the password that is stored in your configuration files. Password encryption allows you to provide a key to encrypt the password. Password hashing provides a one-way hash for passwords, useful for storing passwords of users in the built-in user registry.

You can encode a password by using the `securityUtility` command. In the next step, you encode the admin password and use the encoded version in various configuration files in the course exercises.

__ a. To encode the admin password, `passw0rd`, enter the following command:

```
./securityUtility encode passw0rd
```



Note the encoded password that is displayed. This encoded password is used in the exercises and is placed in various configuration files that you use.

Part 3: Creating the collective controller

Liberty servers can be deployed and administered individually, from a job manager, or by a collective controller as part of a collective. The collective controller provides for a centralized administrative control point to perform operations and provides an agent-less management of Liberty servers. The collective controller also allows you to cluster Liberty servers for high availability and scalability.

In this part of the exercise, you configure the collective controller.

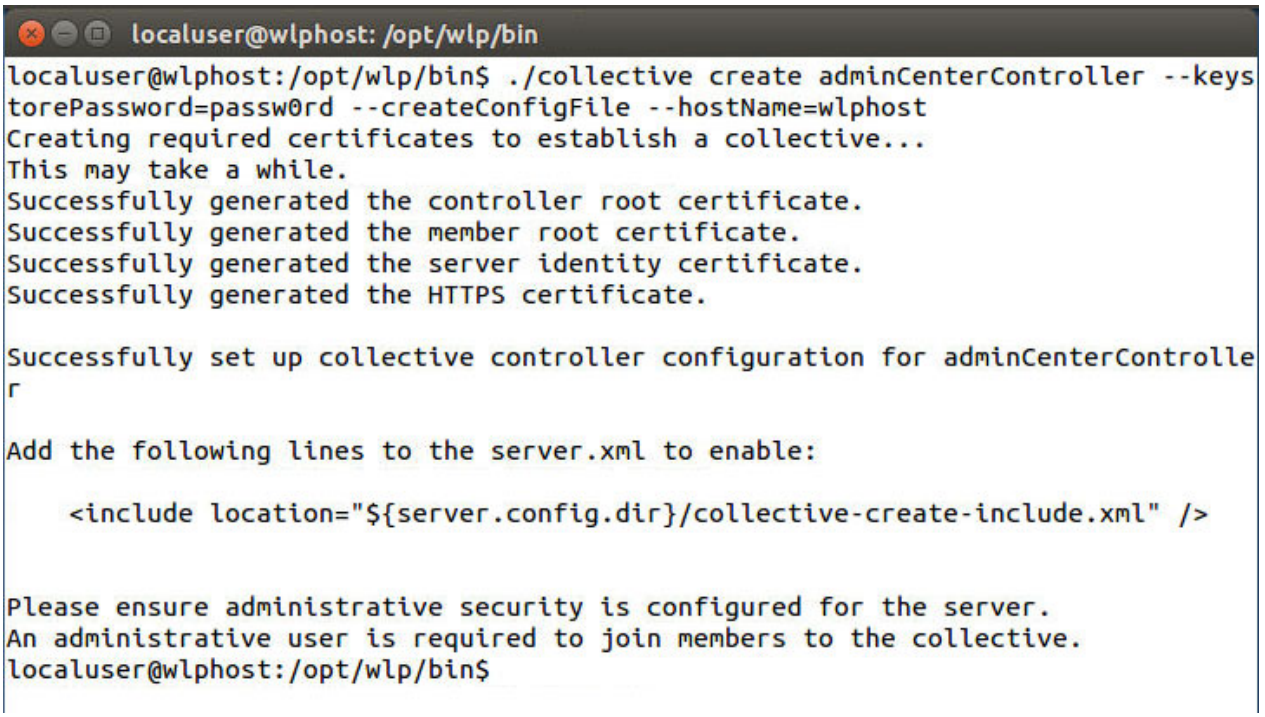
__ 1. Create and configure the adminCenterController, which is the server that you use as the collective controller.

__ a. To create the server adminCenterController, enter the following command:

```
./server create adminCenterController
```

__ b. Create the extra configurations that required to run the adminCenterController as a collective controller. Enter the following command:

```
./collective create adminCenterController --keystorePassword=passw0rd
--createConfigFile --hostName=wlphost
```



Verify that the creation is successful. You can see a note that indicates to add a specific line to the `server.xml` file to include the `collective-create-include.xml` file. You must update these files and tailor them to suit your environment.

__ 2. Course labfiles are provided with tailored scripts and configuration files for the course exercise environment. Next, you review the labfiles Admin configuration files.

__ a. Go to the `/opt/labfiles/Admin` directory.

__ b. Open the `collective-create-include.xml` file and examine the default settings. Open the file by using an editor such as vi or gedit.

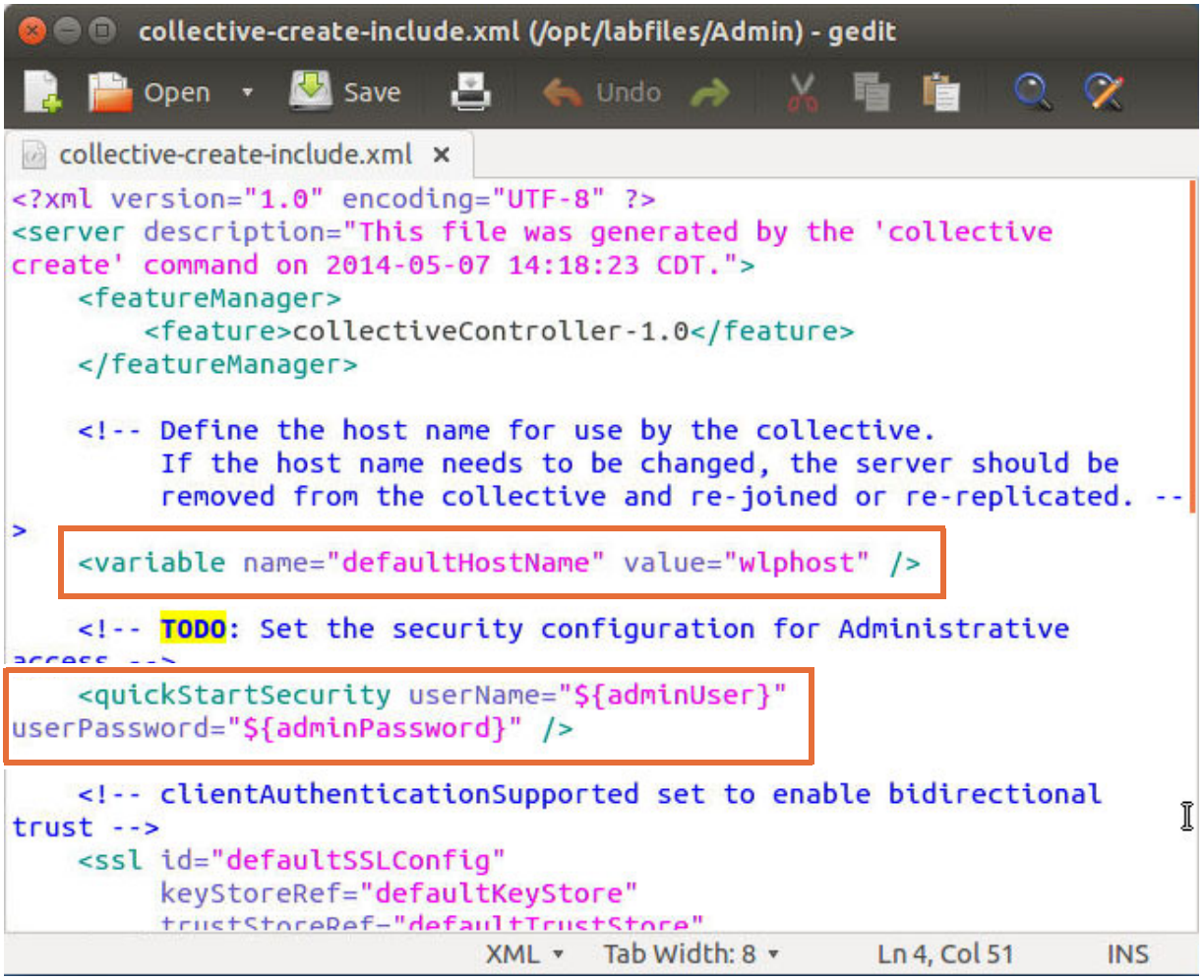
Hint

To use gedit, complete one of the following steps:

- Enter the command: `gedit <filename>&`
- Open the File Browser, change to the directory that contains the file, and double-click the file name.

__ c. Look for the following entires:

```
<variable name="defaultHostName" value="wlphost" />
<quickStartSecurity userName="${adminUser}"
userPassword="${adminPassword}" />
```



You can see that the file has defined a variable of **defaultHostName** with the value `wlphost`. Variables for **userName** and **userPassword** are also noted in the file. However, the variables **adminUser** and **adminPassword** are resolved in the `bootstrap.properties` file.

Information

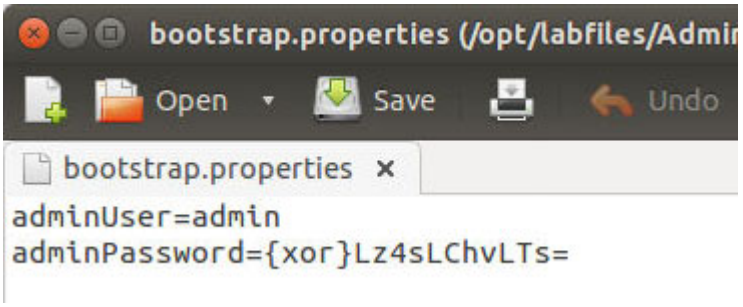
Variables can be used in the configuration of a Liberty server to avoid hardcoding values that might change as the server is reused in different environments. Variables can be defined in either the server configuration file or in the `bootstrap.properties` file. Changes in the server configuration file, `server.xml`, do not require a server restart to take effect. Changes that are made in the `bootstrap.properties` file do require a server restart for the changes to take effect.

It is recommended that variables for a particular server, such as port numbers, be specified in the `bootstrap.properties` file, allowing the `server.xml` file to be shared across multiple servers.

Values that are shared across servers are better defined in a shared xml file that can be included in the `server.xml` of a particular server.

__ d. Close the file when completed.

__ e. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.



The `bootstrap.properties` file is used to resolve the administrator user and password variables. You can see that the values admin, and the encoded form of adminPassword are defined. This entry matches the encoded password that you noted after entering the `securityUtility` command earlier in the exercise.

__ f. Close the file when completed.

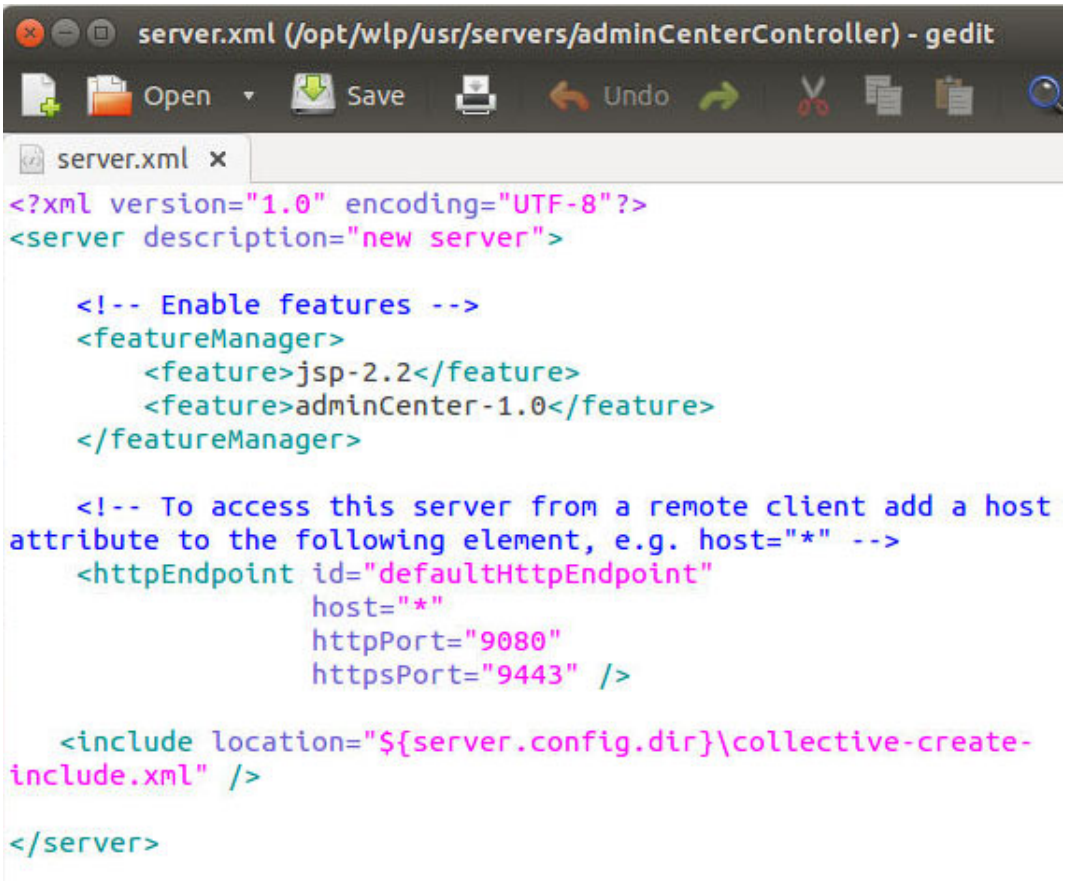
__ 3. Copy the modified files to the directory for the adminCenterController.

__ a. Copy the following files to the `/opt/wlp/usr/servers/adminCenterController` directory:

- `server.xml`
- `collective-create-include.xml`
- `bootstrap.properties`

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.



A few items that you should note in the file:

- The Admin Center feature is added by using the following entry: `<feature>adminCenter-1.0</feature>`
- A host name of "*" is added to allow remote access from all interfaces.
- The file contains an include for the `collective-create-include.xml` file. The `collective-create-include.xml` file is required to enable the server to function as a collective controller. It is the same as the version created by using the collective create command, except for the variables that are introduced to resolve host name, and administrator user and password.

__ d. When completed, close the file.

__ 4. Start the collective controller.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To start the collective controller, enter the following command:

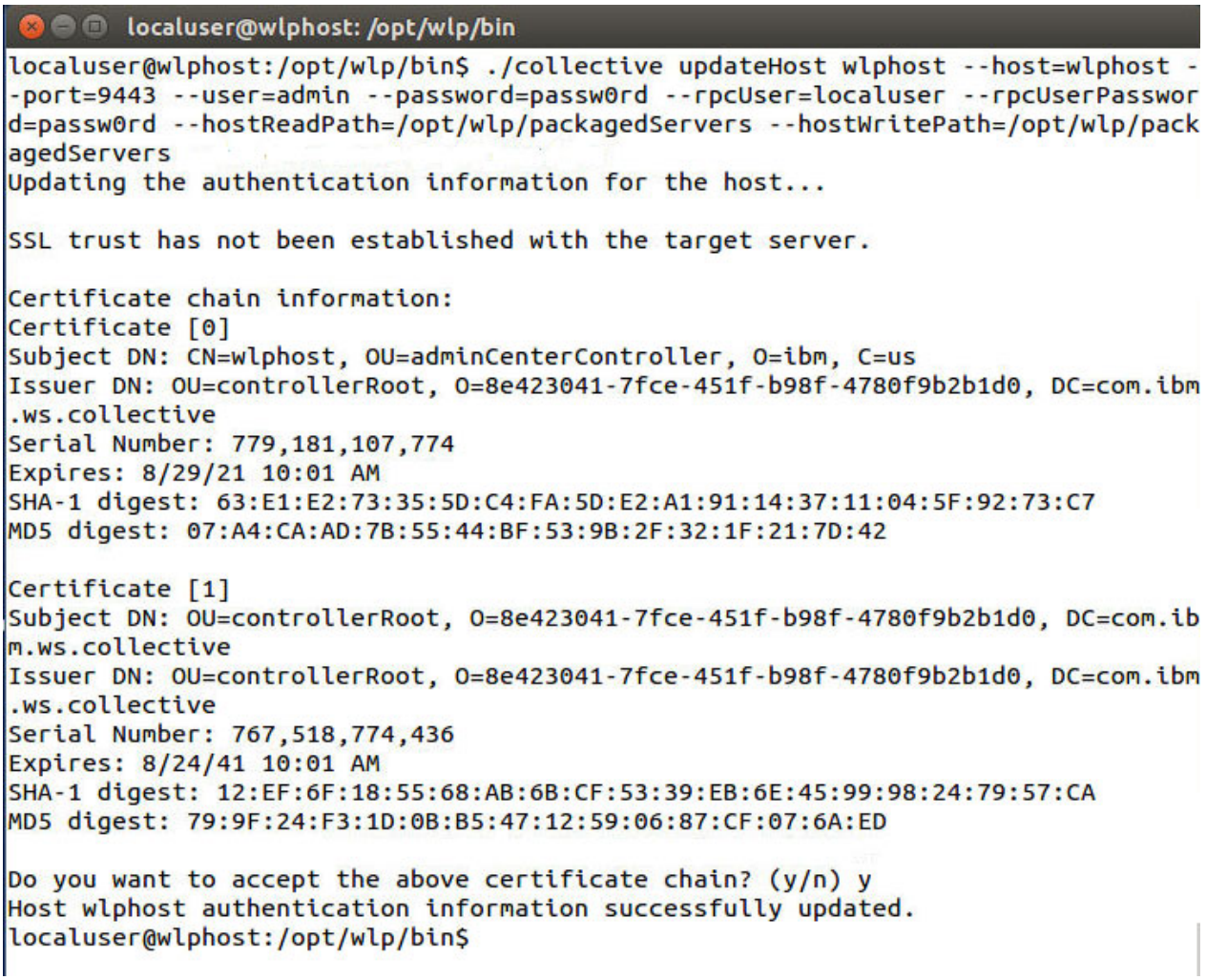
```
./server start adminCenterController
```

__ c. Update the controller with more parameters that are required when you deploy a packaged server. In addition to the Liberty admin user and password, you provide details on the host computer user and password. The option `rpcUser` is the user ID that you use to log in to the operating system, and `rpcUserPassword` is the corresponding password.

To update the controller, enter the following command:

```
./collective updateHost wlphost --host=wlphost --port=9443 --user=admin
--password=passw0rd --rpcUser=localuser --rpcUserPassword=passw0rd
--hostReadPath=/opt/wlp/packagedServers
--hostWritePath=/opt/wlp/packagedServers
```

When prompted, accept the certificate chain.



__ d. Verify that the host information successfully updated.

Information

To transfer files to and from a host, you must specify host read and write paths. The `hostReadPath` specifies the directories that the collective controller can read. The `hostWritePath` specifies the directories to which the collective controller can write. Paths that the `hostWritePath` defines are also readable.

__ e. Minimize the terminal window.

Part 4: Working with the Admin Center

The Liberty Administrative Center (Admin Center) feature is available for Liberty, which provides a web-based graphical interface for Liberty servers and resource management. The Admin Center can be used to administer Liberty servers, applications, clusters, and hosts from a web browser on a smartphone, tablet, or computer. Admin Center offers the ability to view details about and perform operations (start, stop, restart, add and remove metadata, and enable and disable maintenance mode) on resources within the collective. It also offers the ability to edit server configuration files to view bookmark information, to add custom tools to monitor server resources, and to deploy server packages on hosts within the collective.

In this part of the exercise, you work with the Admin Center.

__ 1. Start the Admin Center console.

__ a. Open the Firefox web browser and go to the following website:

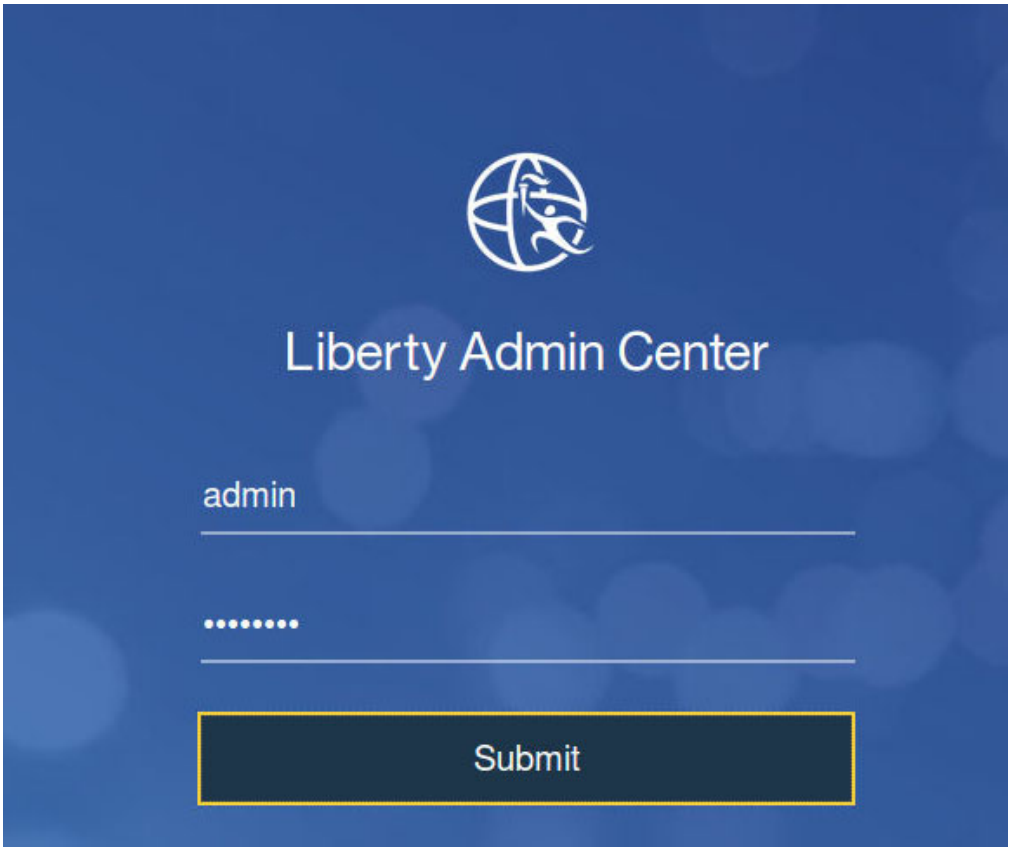
`http://wlphost:9080/adminCenter`

__ b. When prompted that the connection is not secure, click **Advanced**. Then, add a security exception by clicking **Add Exception**. Finally, click **Confirm Security Exception**.

Hint

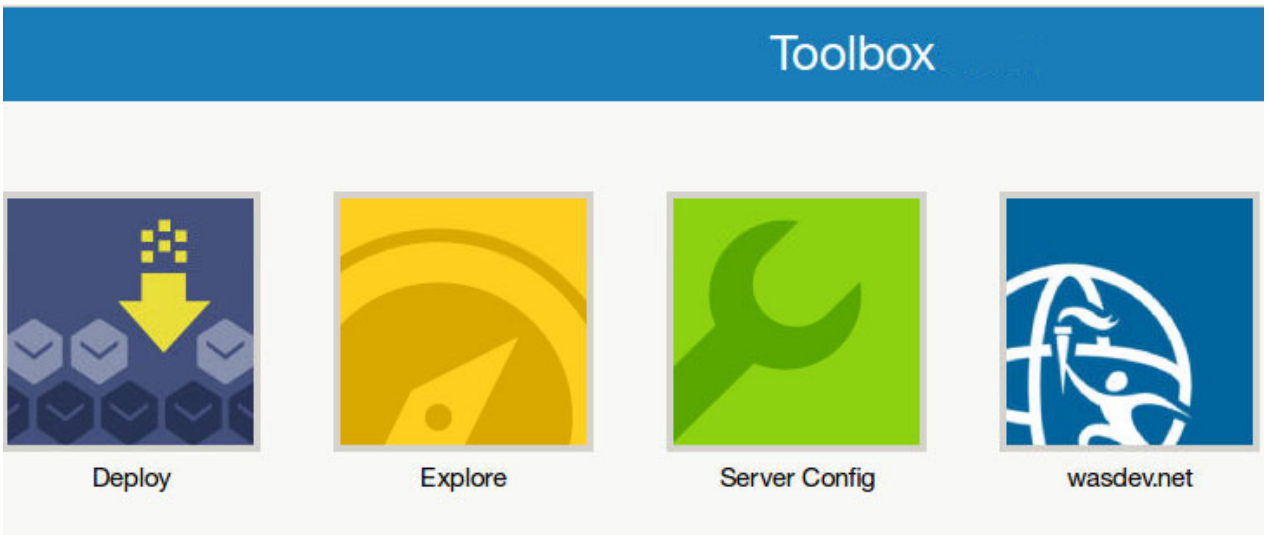
Since the Liberty Admin Center is used multiple times throughout the exercises, it might be a good idea to create a bookmark to the URL.

__ c. In the login area, enter `admin` as the User Name and `passw0rd` as the Password and click **Submit.**



__ 2. Explore the Admin Center.

__ a. You are placed on the Admin Center main page, which shows the Toolbox.

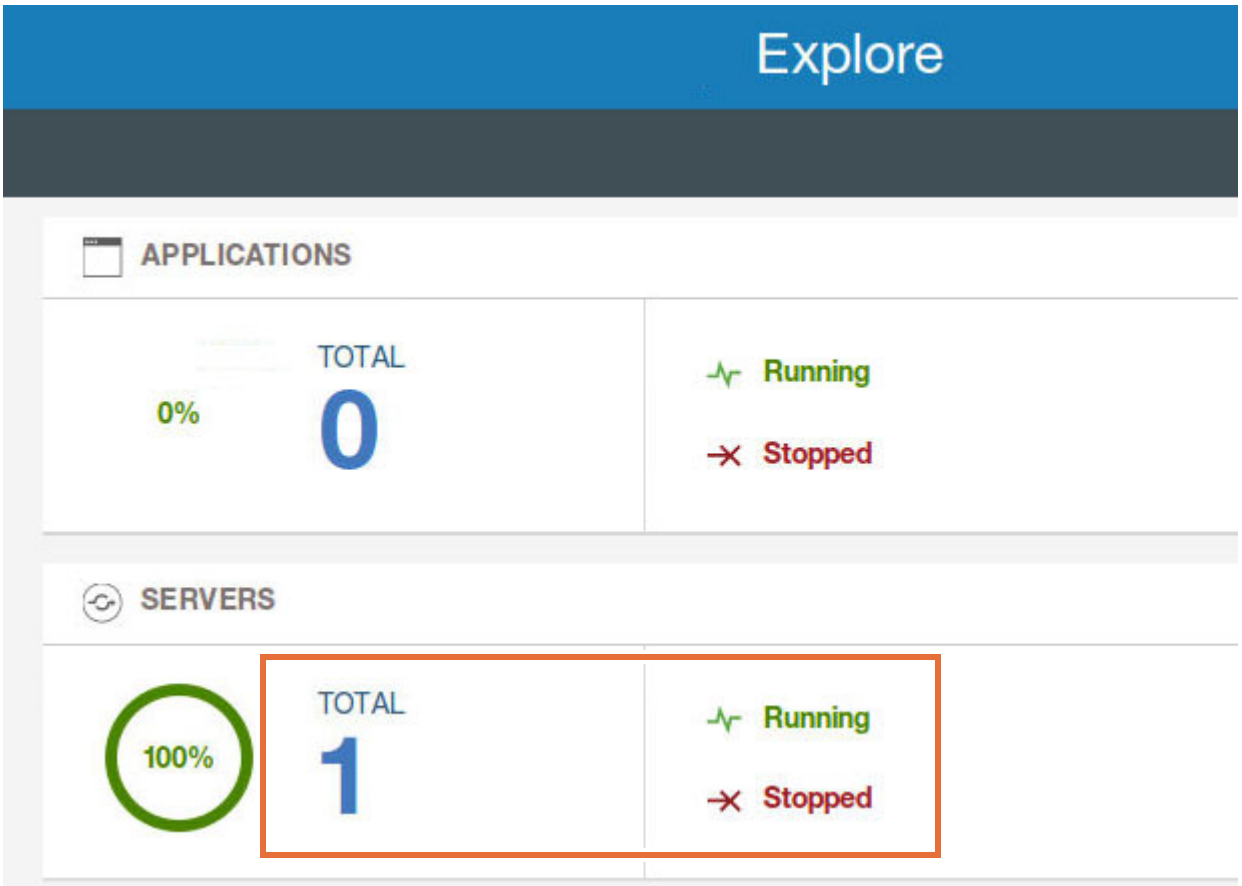


Information

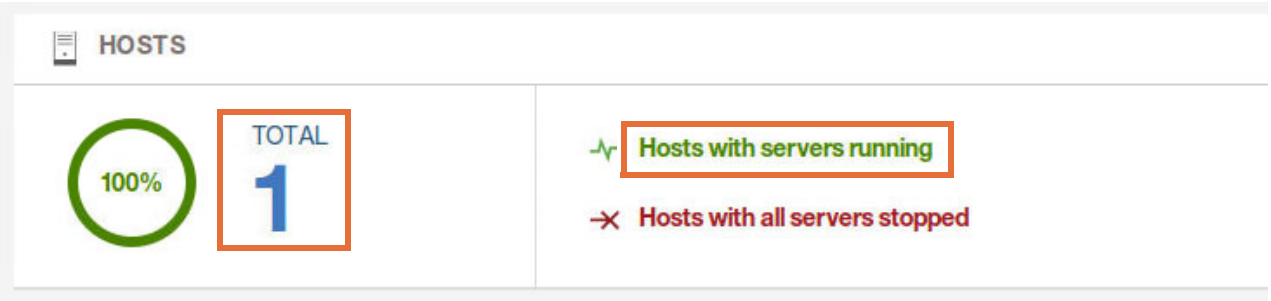
When you first log in to Admin Center, the Toolbox contains the Server Config and Explore tools plus a bookmark to WASdev.net. The Toolbox also has the Deploy tool if the server that is hosting the Admin Center is a collective controller. You can edit the Toolbox contents by adding tools from the Tool catalog or by adding bookmarks. A tool is a web application that completes a particular task and is typically provided by the product. A bookmark is a user-added link to any site. You can further customize your Toolbox by rearranging icons, filtering Toolbox contents, or editing preferences.

__ b. Click **Explore**. This action places you on the Explore page, or dashboard.

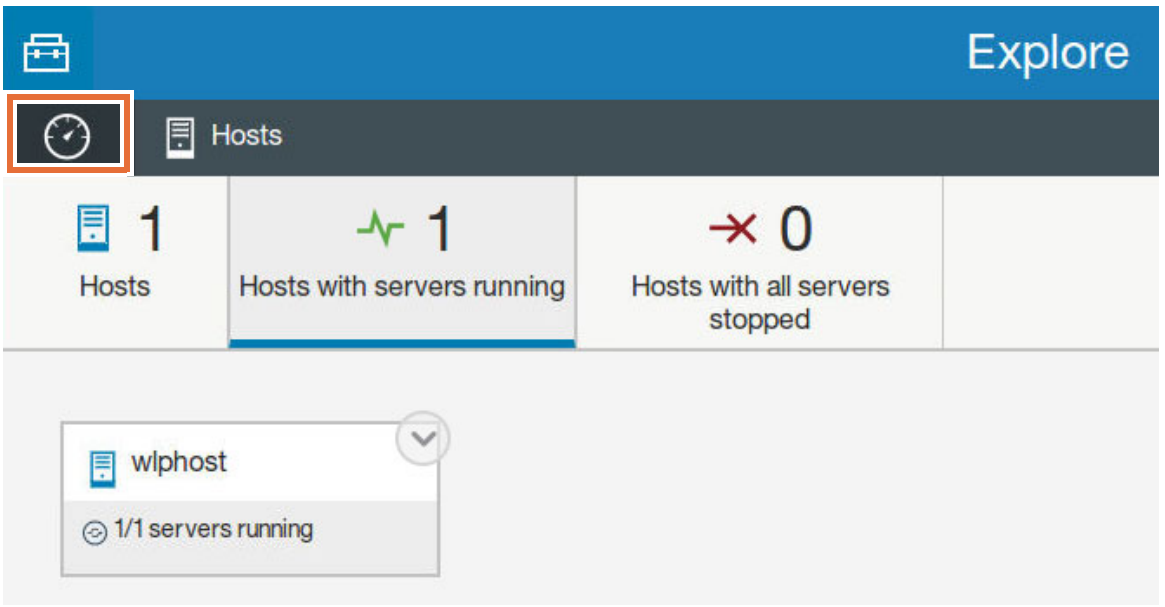
__ c. You can use the Explore tool to view and manage resources in the Liberty topology. On this page, you get a summary of all resources on the Explore tool dashboard. Go to the SERVERS section. You can see that one application server running, which is the adminCenterController.



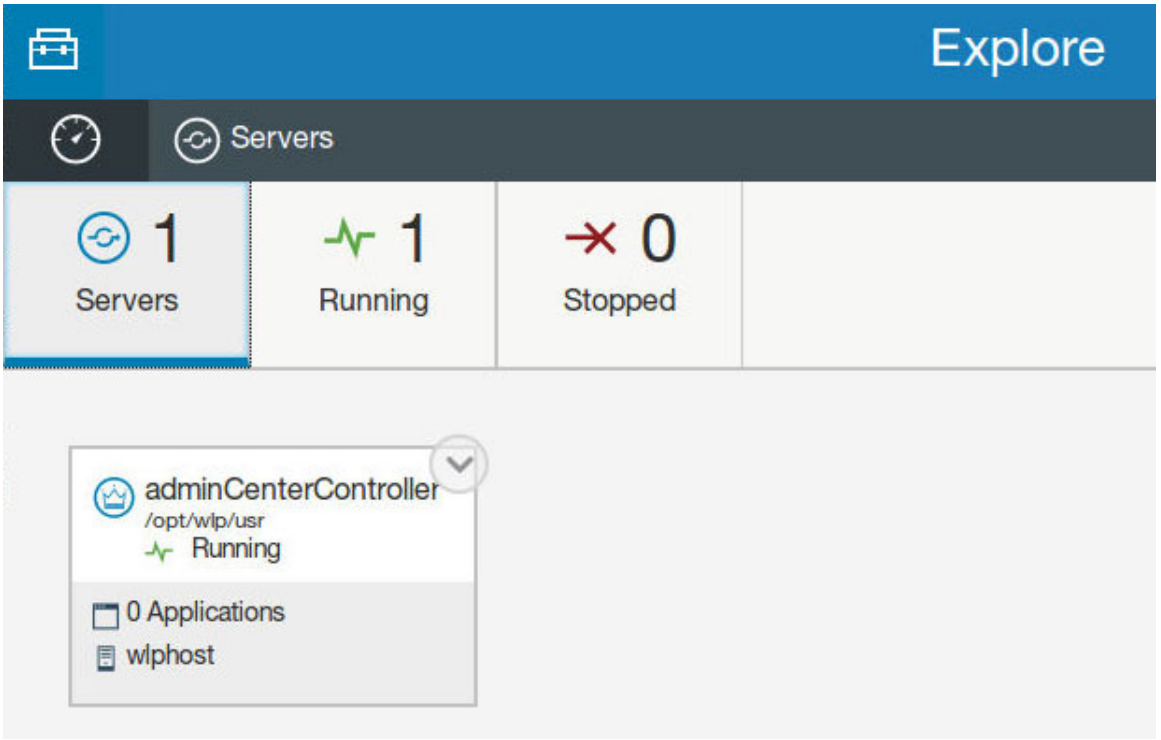
__ d. Go to the HOSTS section. You can see that one host that is being managed. Under HOSTS, click the **Total 1** or **Hosts with servers running**.



__ e. From here, you can view details about a resource. You can see details about the hosts. To go back, click the **Dashboard** icon.



__ f. Under SERVERS, click **Total 1** to examine the server details. You can see that the running server is the adminCenterController.



__ g. Click the **Dashboard** icon. Feel free to continue exploring the Admin Center.

__ h. When completed, click the **Toolbox** icon to go back.

__ i. Minimize the Admin Center browser window.

Part 5: Preparing and deploying a packaged server

A packaged server contains a `server.xml` file, an application, and optionally the Liberty profile run time. The Admin Center currently only supports packaged server deployment with packages that also contain the run time.

After you define the Liberty collective and create the server package, you can use the Admin Center Deploy tool to install the server package on hosts within the collective. In this part of the exercise, you create a packaged server and deploy it by using the Admin Center.

__ 1. Copy and examine the snoop_1 files.

__ a. In a terminal window, go to the `/opt/labfiles/Admin` directory.

__ b. Copy the `snoop_1` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

```
cp -r snoop_1 /opt/wlp/usr/servers/
```

__ c. Go to the `/opt/wlp/usr/servers/snoop_1` directory.

__ d. The directory contains three files and the dropins directory. Open the `server.xml` file by using an editor such as vi or gedit. You can see that the file includes the following features:

- **collectiveMember-1.0**: The feature indicates that the controller manages the server.
- **clusterMember-1.0**: This feature indicates that the server is to be a member of a cluster.

server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>collectiveMember-1.0</feature>
        <feature>clusterMember-1.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host
attribute to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint"
        httpPort="${httpPort}"
        httpsPort="${httpsPort}" />


    <clusterMember name="snoopCluster" />


</server>
```

__ e. Notice that variables are also used for **httpPort** and **httpsPort**. Close the file when completed.

__ f. Open the `bootstrap.properties` file by using an editor such as vi or gedit. In this file, you can see that the variables for **httpPort** and **httpsPort** are defined. Close the file when completed.

__ g. Open the `admin-metadata.xml` file by using an editor such as vi or gedit. This file provides more information that helps you understand the environment, and it is important as the complexity of the environment increases. Later in the exercise, you see how the information that is provided in this file is used and examined within the Admin Center.

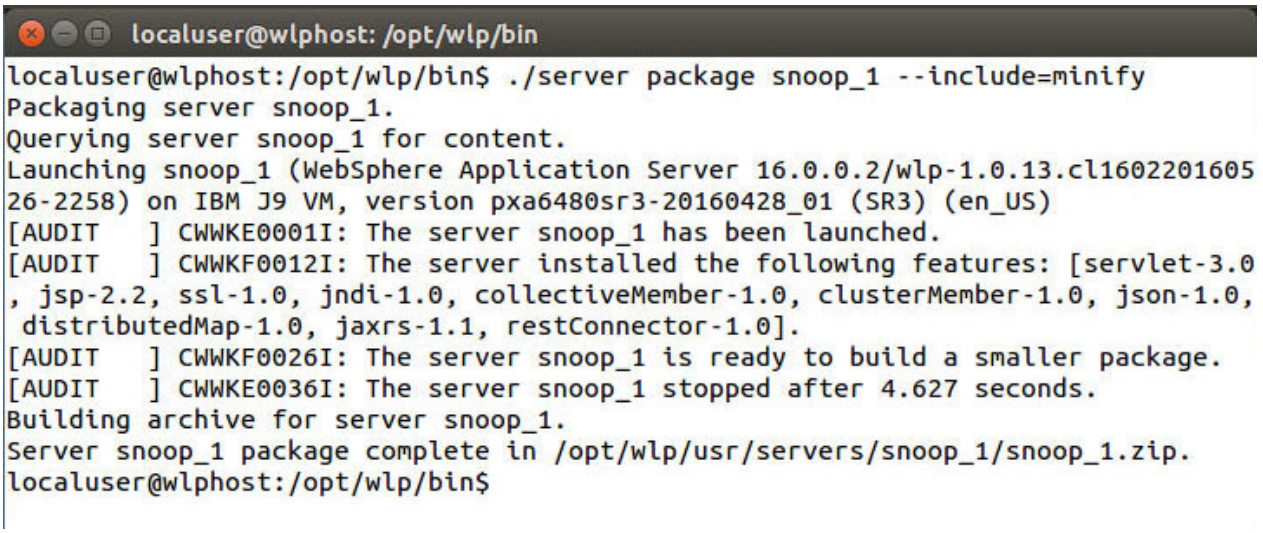
__ h. Close the file when completed.

__ 2. Create a packaged server. From the command line, you can create a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server.

__ a. In the terminal window, go to the `/opt/wlp/bin` directory.

__ b. To create the packaged server snoop_1, enter the following command:

```
./server package snoop_1 --include=minify
```



Wait until you see that the package is complete. You can also see that this command creates a file, `snoop_1.zip`, in the `snoop_1` directory.

Information

The `minify` option indicates to include only the required features in the package, rather than the entire Liberty profile run time. The command packages only those parts of the runtime environment, and files in the `${WLP_USER_DIR}` directory, that is required to run the server. This option significantly reduces the size of the resulting archive.

__ c. Minimize the terminal window.

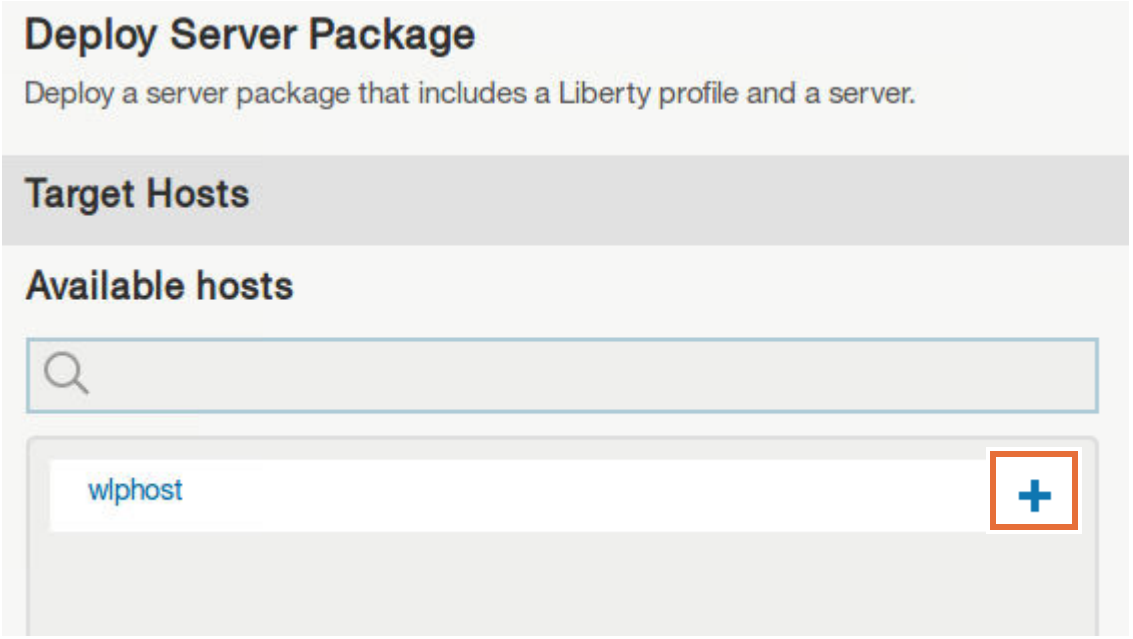
__ 3. Deploy a packaged server by using the Admin Center.

__ a. Maximize the Admin Center browser window.

__ b. In the Admin Center, click **Deploy**.

__ c. Go to the Available hosts section.

__ d. Select the host that you want to deploy the package to. Next to wlphost, click **+**.



You can see that the host, wlphost, is moved to the Selected hosts section.

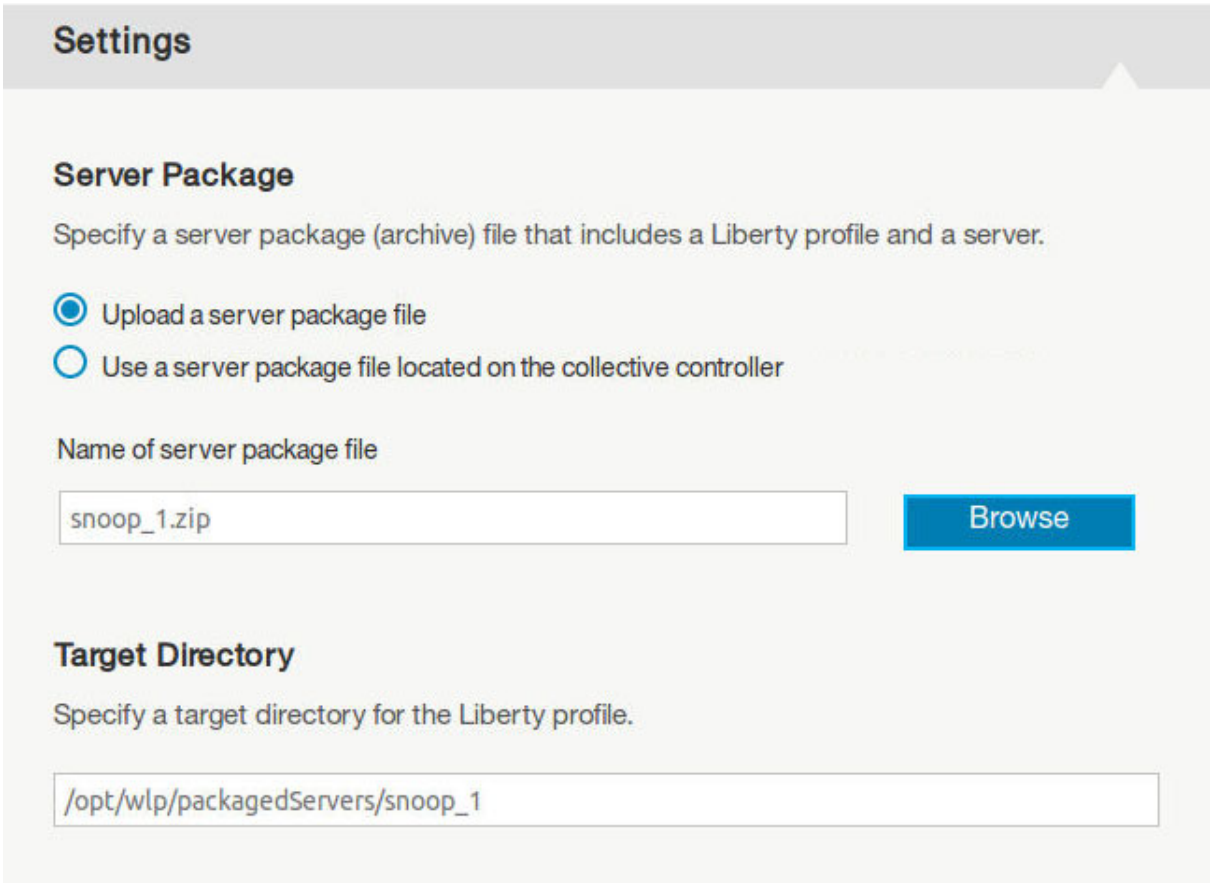
__ e. Go to the Settings section.

__ f. Under Server Package, verify that the option **Upload a server package file** is selected.

__ g. Click **Browse,** select `/opt/wlp/usr/servers/snoop_1/snoop_1.zip`, and click **Open**.

__ h. In the Target Directory field, enter the following directory:

`/opt/wlp/packagedServers/snoop_1`

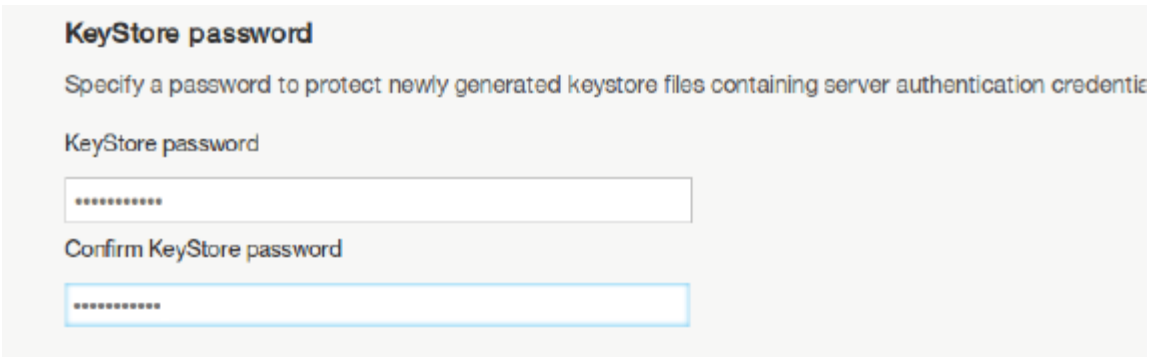


Hint

Recall that earlier in the exercise, you entered a collective `updateHost` command to inform the controller what directory you are able to deploy to.

__ i. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`



__ j. Go to the Remote Management Credentials section.

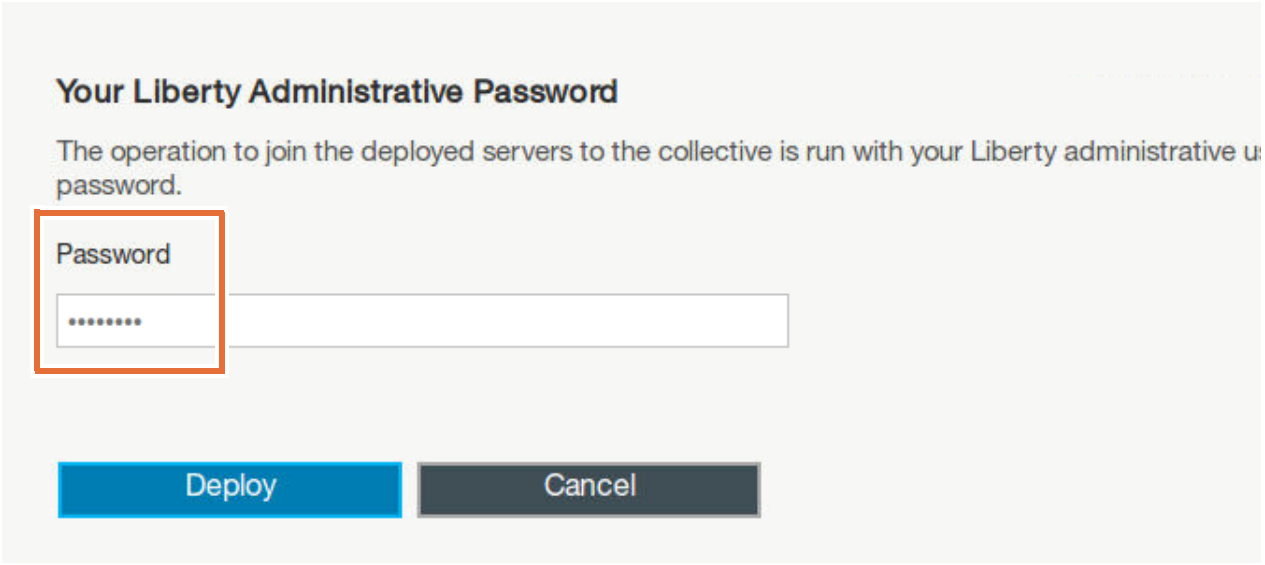
__ k. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

Hint

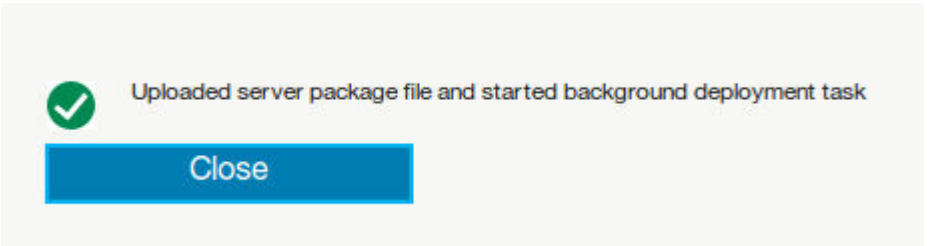
Recall that earlier in the exercise, you configured how the collective controller connects to each target host by using the collective `updateHost` command. This option allows you to choose whether to use SSH or user ID and password to manage your application server by using configurations at the controller, instead of at the application server.

__ l. Go to the Your Liberty Administrative Password section.

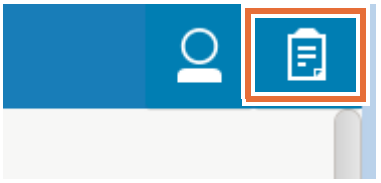
__ m. In the **Password** field, enter: `passw0rd`.



__ n. At the bottom of the page, click **Deploy.** Wait for the message that the file is uploaded and a background deployment task is started.



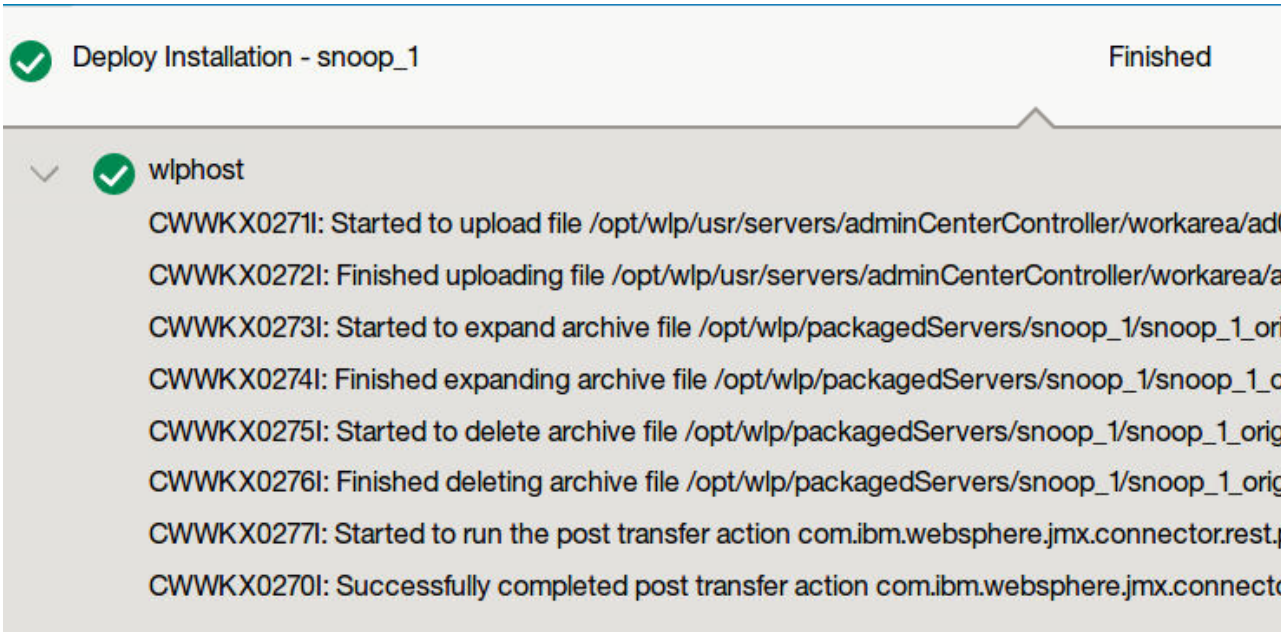
__ o. To see the Task History, click the **Background Tasks** icon at the upper-right corner of the page.



__ p. Click **Task Details and History** to see the status of the deployment.

__ q. The green check mark and green progress bar indicate a successful deployment.

__ r. To examine the details, click **Deploy Installation - snoop_1**. Then, click **wlphost**.



Troubleshooting

If you get the error message: "`CWWKX7204E: Cannot connect to host <host name> with the credentials provided`", verify that you entered the correct user ID and password. Examine the `collective updateHost` command that you entered earlier in the exercise.

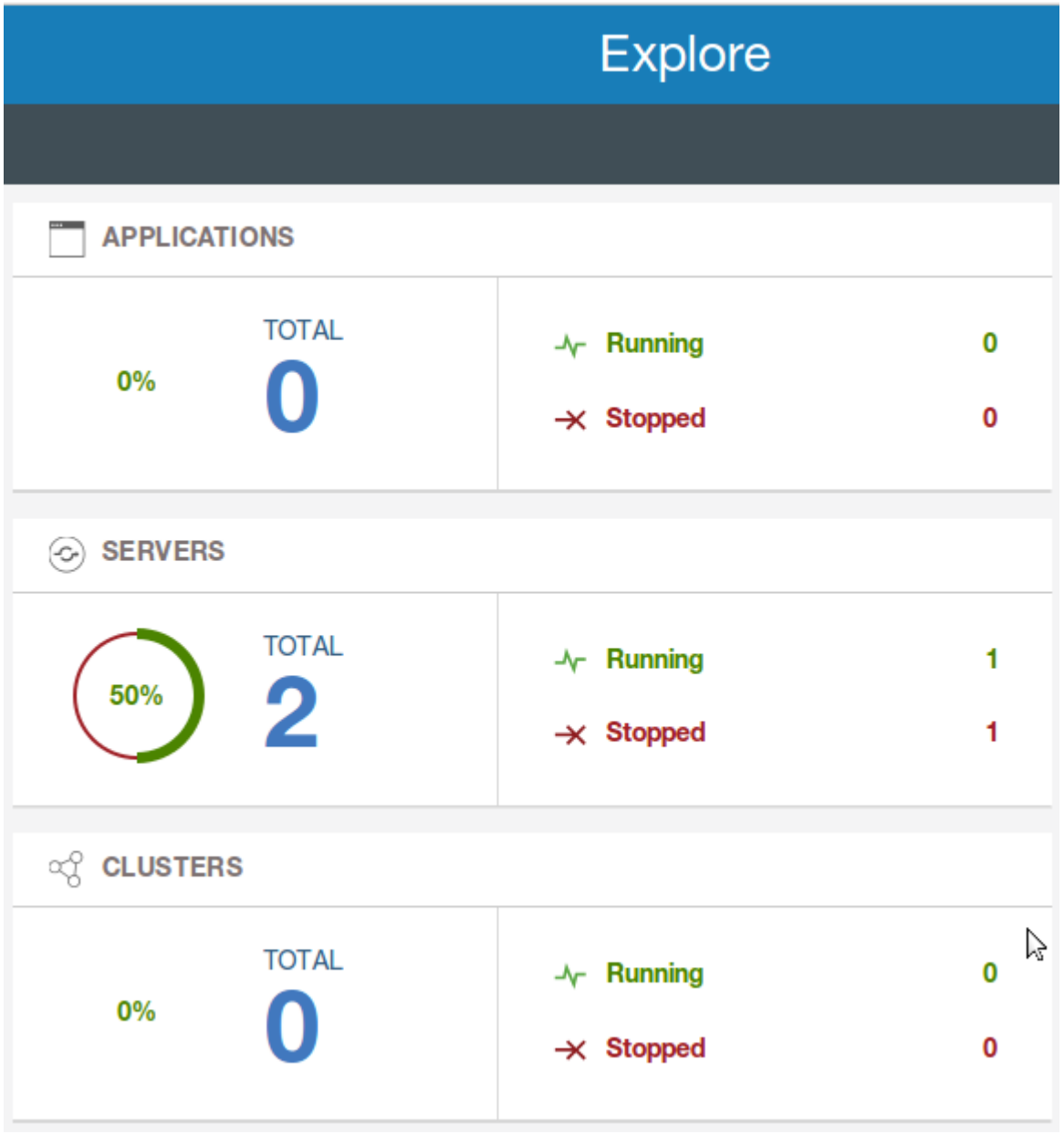
If you get the error message `CWWKX0262E`, verify that you specified the correct `hostReadPath` and `hostWritePath` options when running the `collective updateHost` command. You can run the command again with the correct values. Also, verify that you specified `/opt/wlp/packagedServers/snoop_1` as the destination directory.

__ 4. Examine and work with the server.

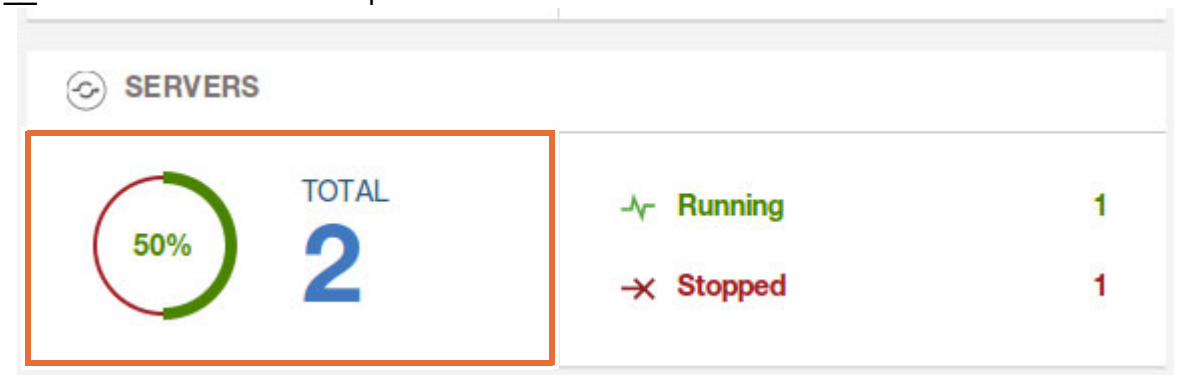
__ a. Click **Toolbox**.

__ b. Click **Explore**.

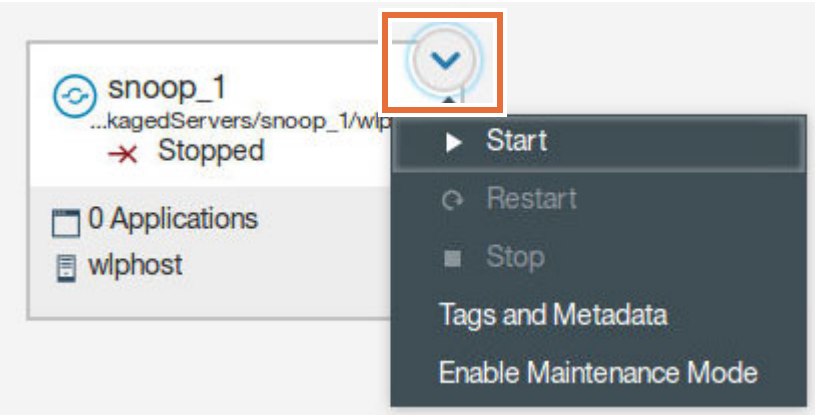
__ c. Go to the SERVERS section. Notice the two servers, one running and one stopped. However, no cluster is detected yet.



__ d. Click the **SERVERS** pane.

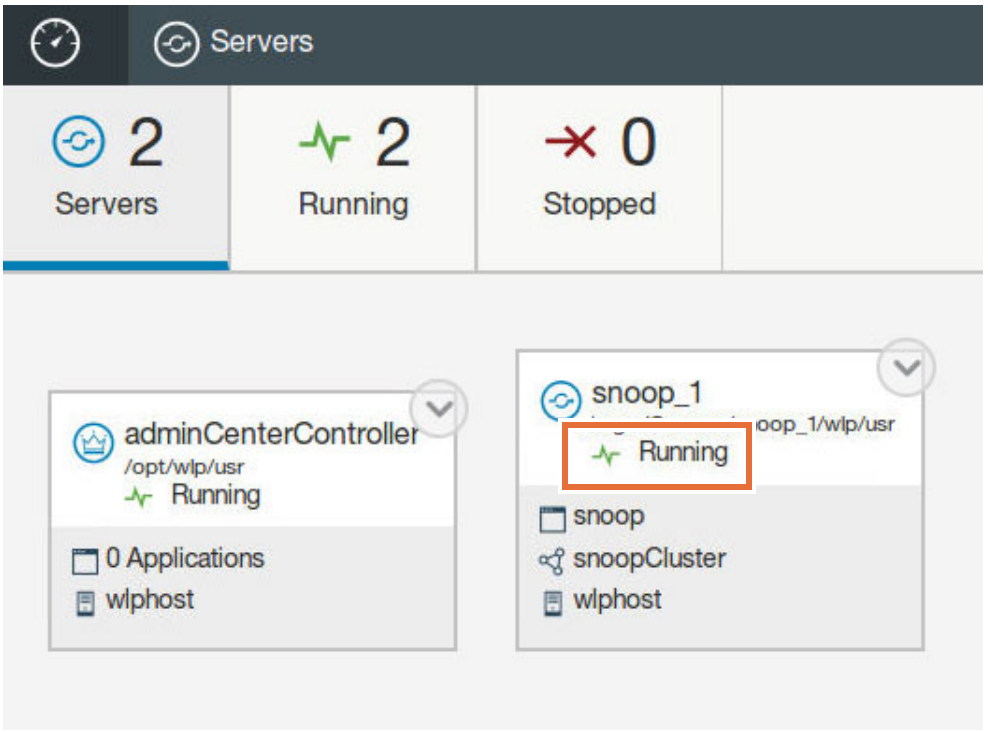


__ e. You have two servers, adminCenterController and snoop_1. Start snoop_1. Click the **Actions** icon (down-arrow) in the upper right of the snoop_1 server box.



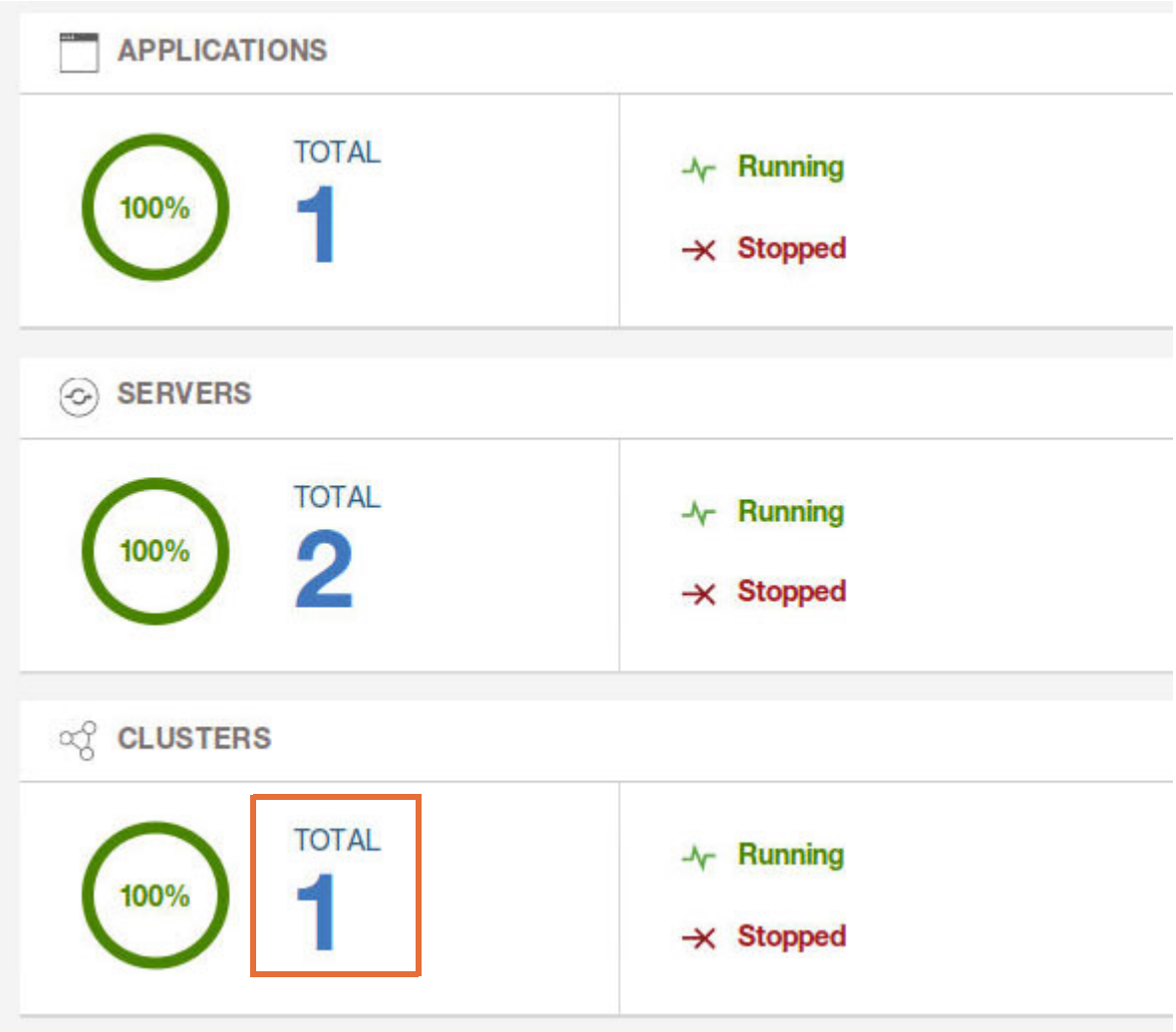
__ f. In the start server snoop_1 window, click **Start**.

__ g. After a few minutes, the status of the server changes to **Running**.

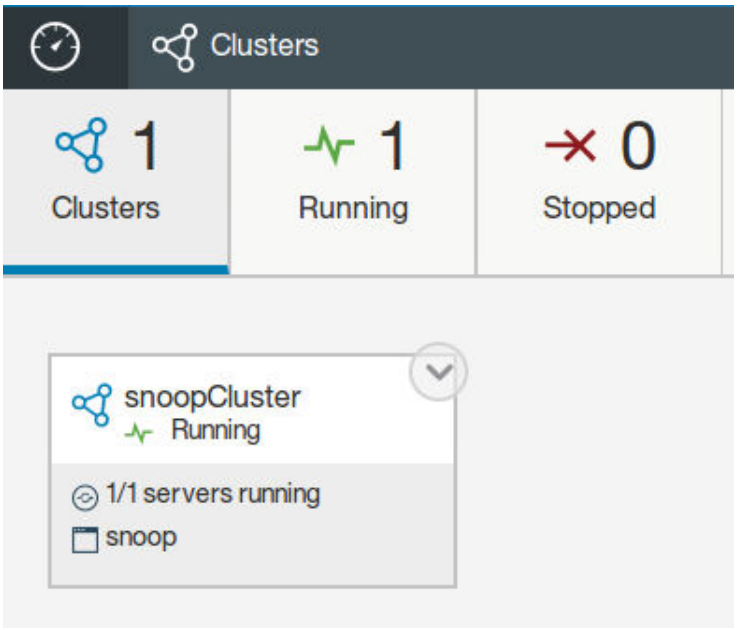


__ h. Click the **Dashboard** icon to go back to the dashboard.

__ i. Go to the CLUSTERS section. You can see that a cluster is now detected and running.



__ j. Click the **CLUSTERS** section. You can see that the cluster, snoopCluster, is running and it includes one server. The cluster also includes one application, snoop, which is running.



__ k. Click the **Dashboard** icon to go back to the dashboard.

__ l. Feel free to explore the Admin Center on your own. Examine how to start and stop clusters or applications. When completed, minimize the Admin Center.

__ 5. Examine the directory structure.

__ a. Go to the terminal window.

__ b. Go to the `/opt/wlp/packagedServers/snoop_1/wlp/usr/servers/snoop_1` directory.

__ c. List the contents of the directory. The Admin Center creates the `collective-join-include.xml` file, which allows the member to join with the controller.

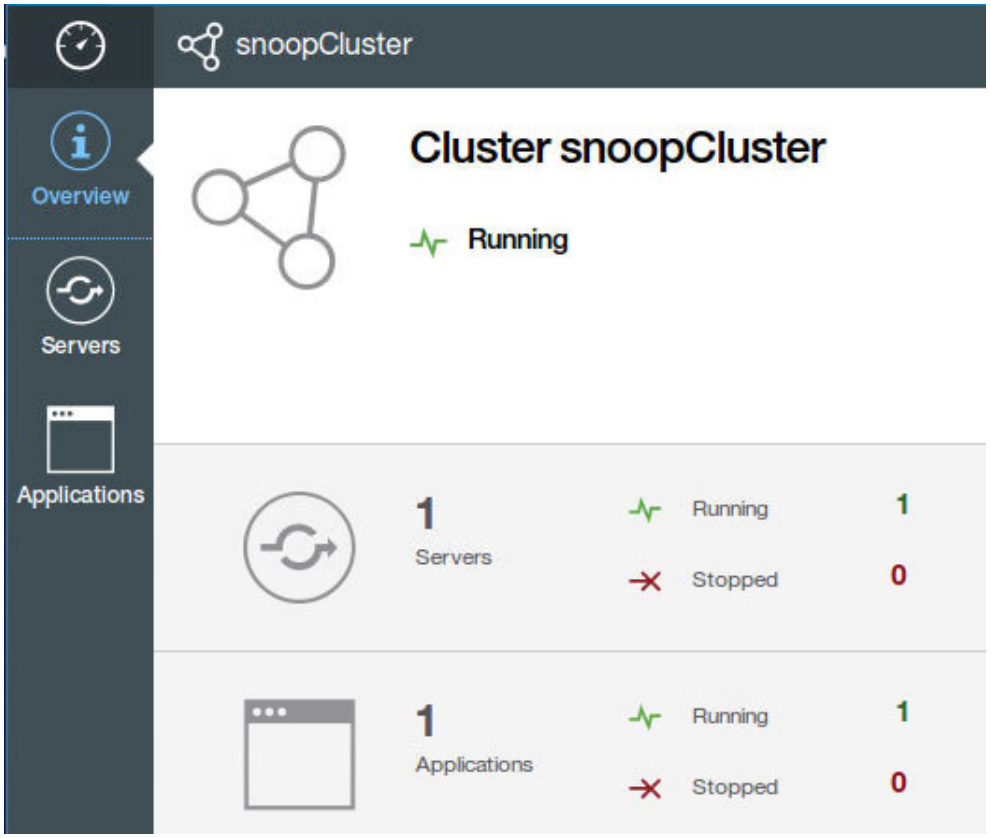
__ 6. Examine the administrative metadata that is defined.

__ a. Maximize the Admin Center.

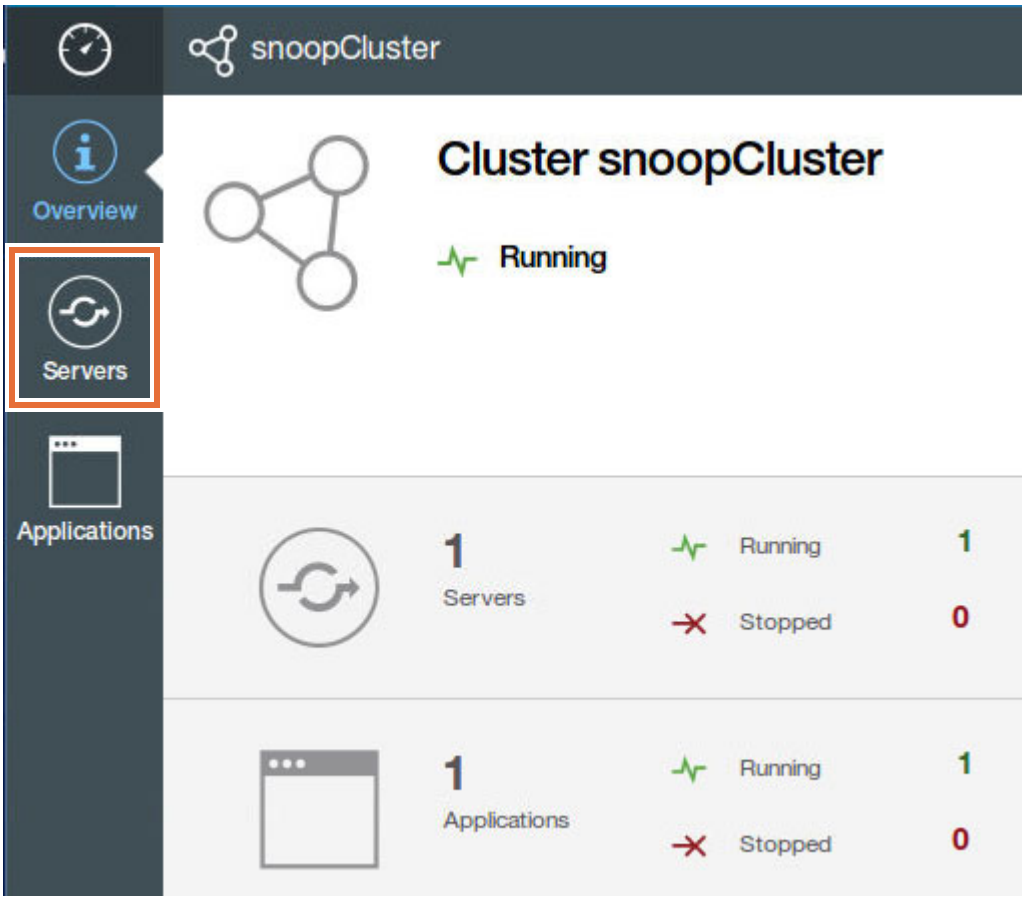
__ b. If needed, go to the Explore pane.

__ c. Click the **CLUSTERS** section.

__ d. Click **snoopCluster**. From here, you get an overview of the cluster. You can see that the snoopCluster is running on one server and it includes one application.

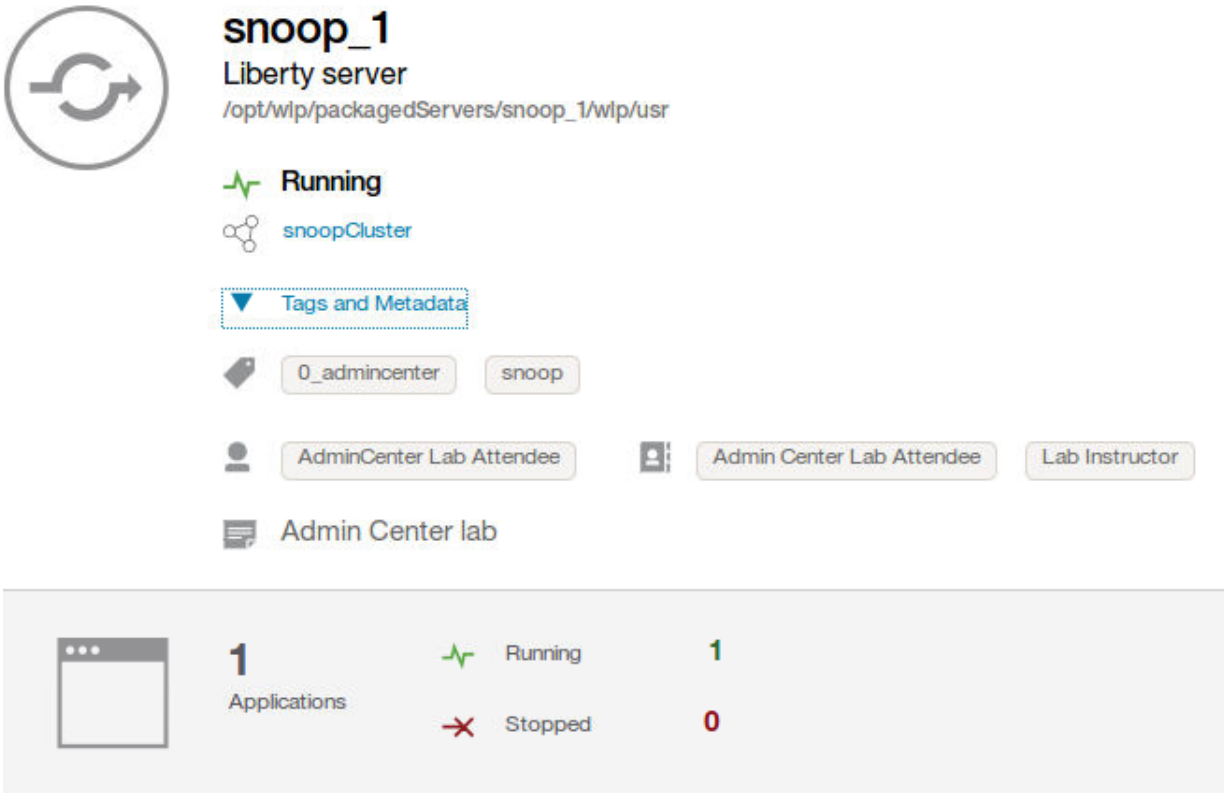


__ e. In the navigation on the left, click **Servers**.



__ f. Click **snoop_1**. From here, you can see details on the server such as status, applications, and more.

__ g. Click **Tags and Metadata**. You can see that the administrative metadata for the server is displayed. Recall this metadata was defined in the `admin-metadata.xml` file.



Information

The information that you provide in the `admin-metadata.xml` file is associated with specific servers and clusters. These include tags, contacts, and owner. The Admin Center displays the administrative metadata that is associated with each server and cluster. You can provide administrative metadata to give you and other administrators more context about the server or cluster that is being managed.

Part 6: Using the Admin Center

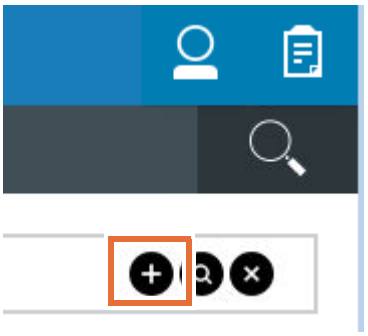
You can use the Explore tool to explore and manage resources in a Liberty topology. If you manage many resources, the search tool in the Explore dashboard can be used to search for resources that fit a particular search criteria. The Explore dashboard also includes a way for you to monitor metrics on your Liberty topology. You can use the Monitor view to track used heap memory, loaded classes, active Java virtual machine (JVM) threads, central processing unit (CPU) usage, and other metrics depending on the resource. The Monitor view shows the metrics graphically in charts and you can customize the view by selecting the charts to show or hide.

In this part of the exercise, you use the search tool and view monitor metrics by using the Admin Center.

__ 1. Work with the Search function in the Explore dashboard.

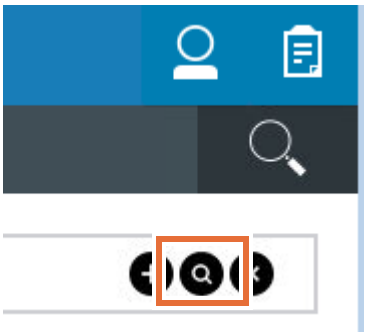
__ a. In the Explore tool dashboard, click the **Search** icon.

__ b. To add search criteria, next to the search field, click the **+** icon.

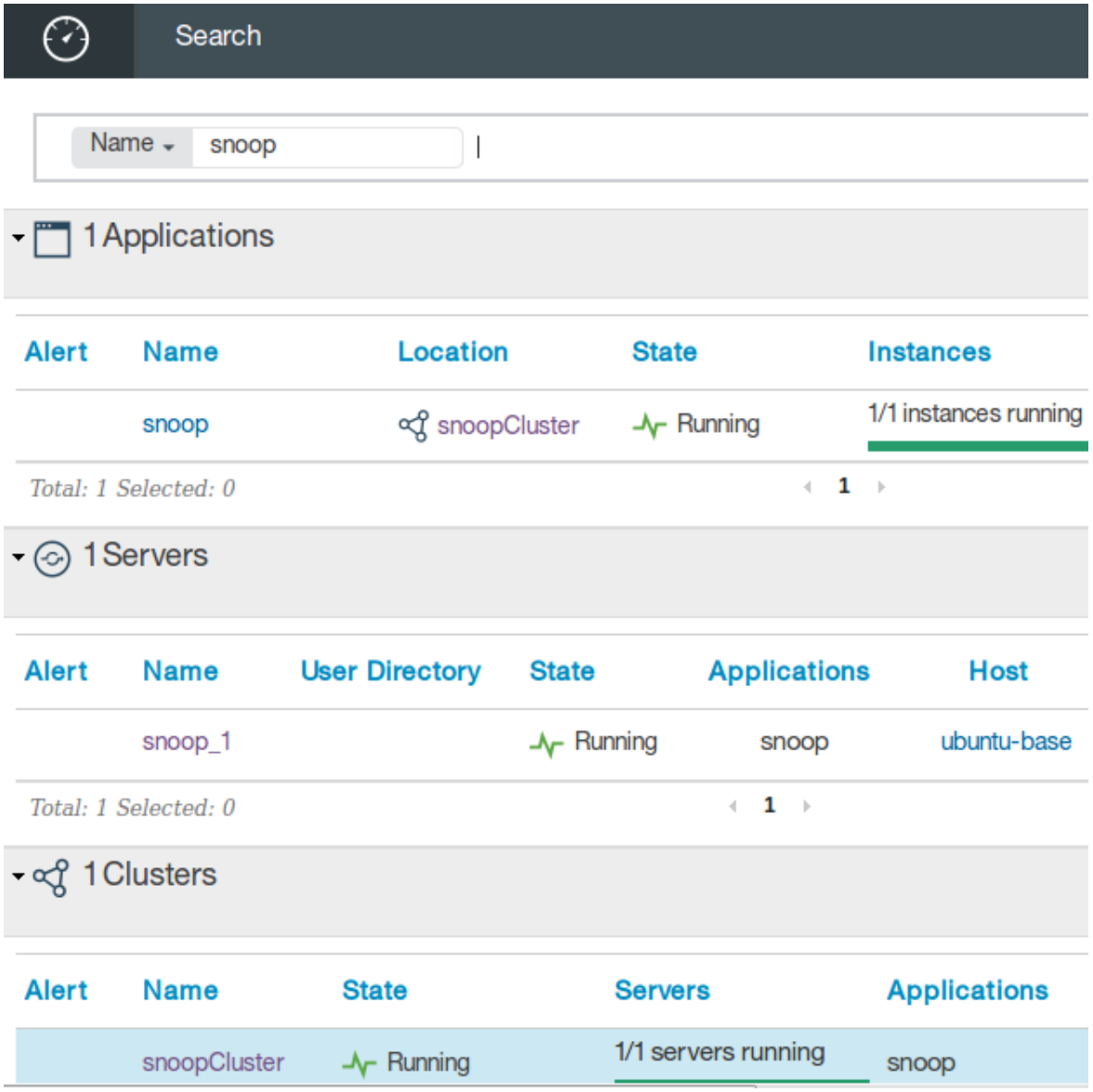


__ c. The initial search criteria is to search by name. In the search field next to **Name**, enter: `snoop`

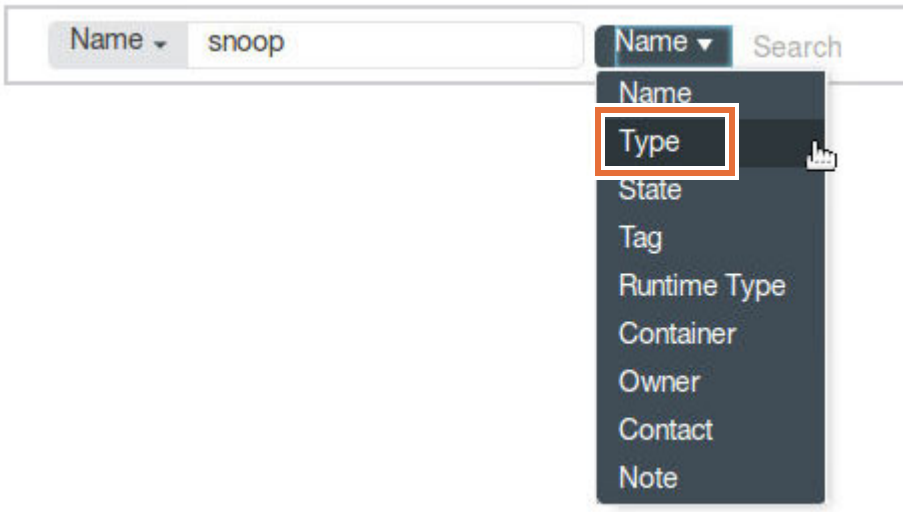
__ d. Click the **Search** icon to begin the search.



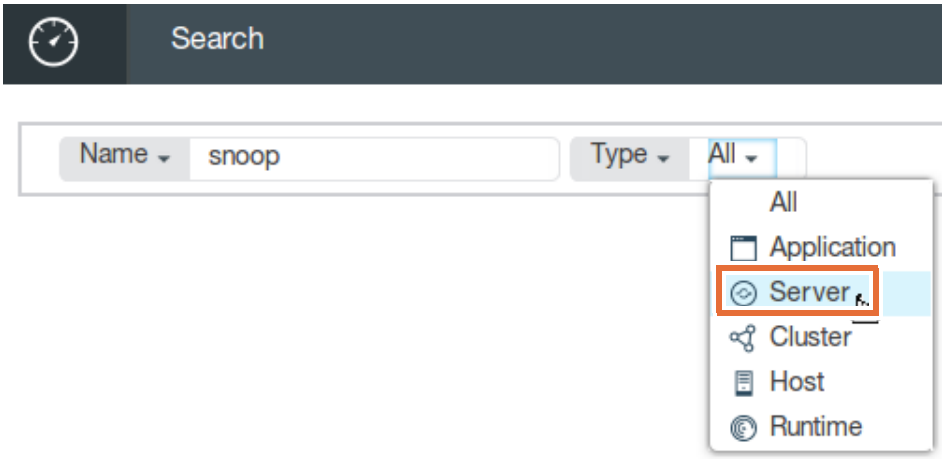
__ e. The results of search indicate one application, one server, and one cluster with a name that contains the substring "snoop".



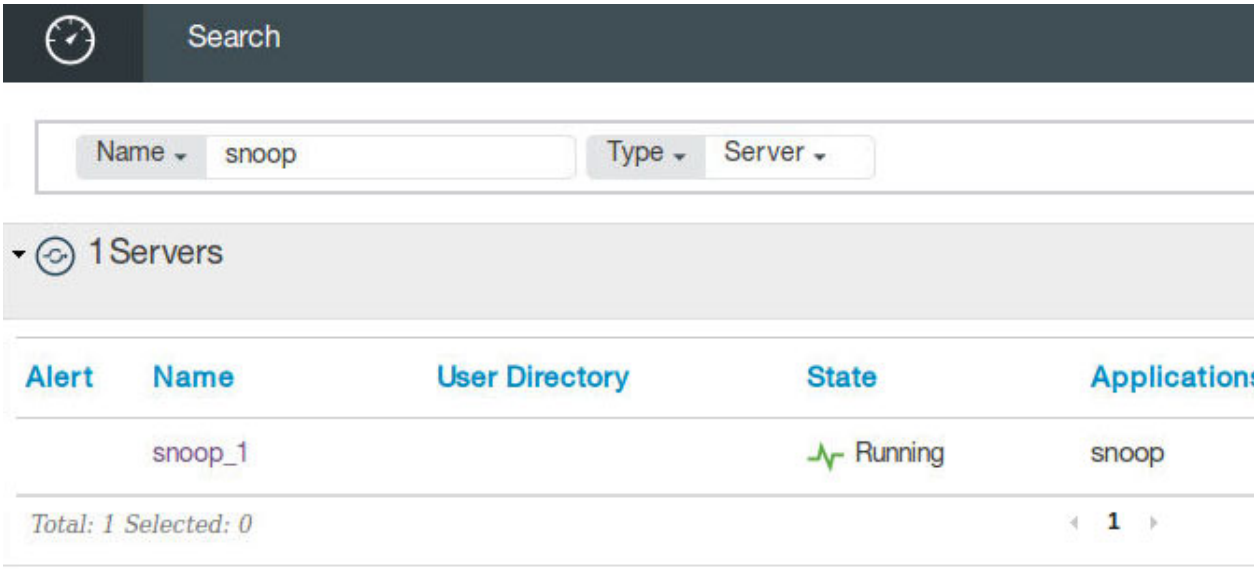
__ f. In the Search field, click the **+** icon and change the search criterion to **Type**.



__ g. Then, from the menu, select **Server**.



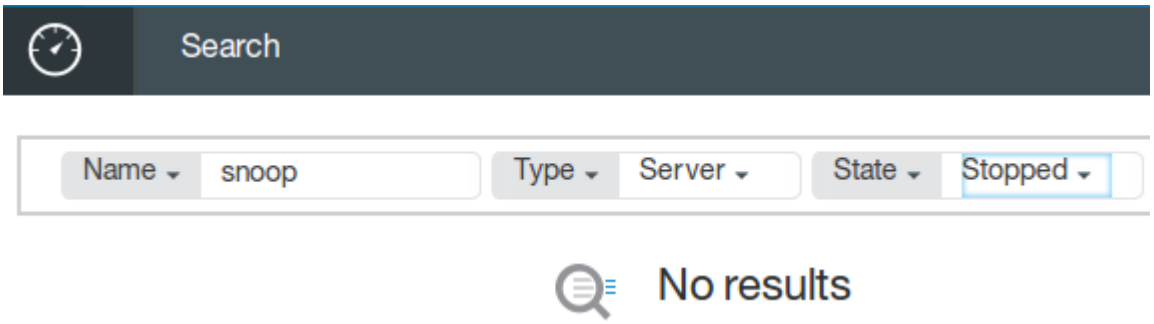
__ h. The results indicate that only one server whose name contains the substring "`snoop`" and the results are displayed.



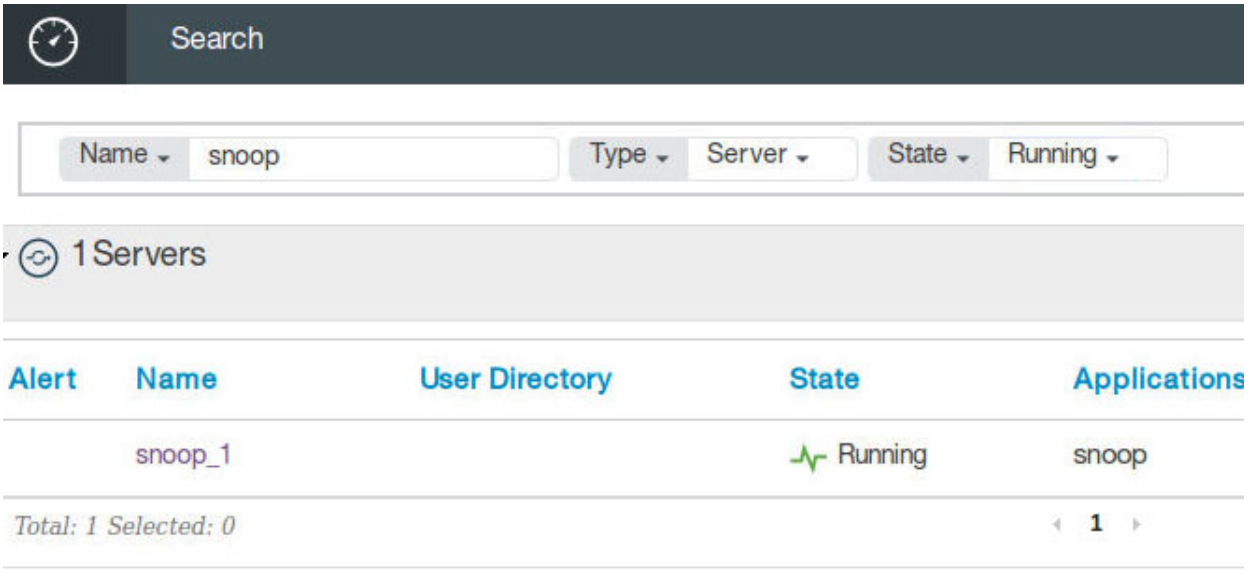
__ i. In the Search field, click the **+** icon and change the search criterion to **State**.

__ j. Then, from the menu, select **Stopped**.

__ k. No stopped servers whose name contains the substring "`snoop`" are found.



__ l. In the search field, change the State to **Running**. The results page displays the running server whose name contains the substring "`snoop`".



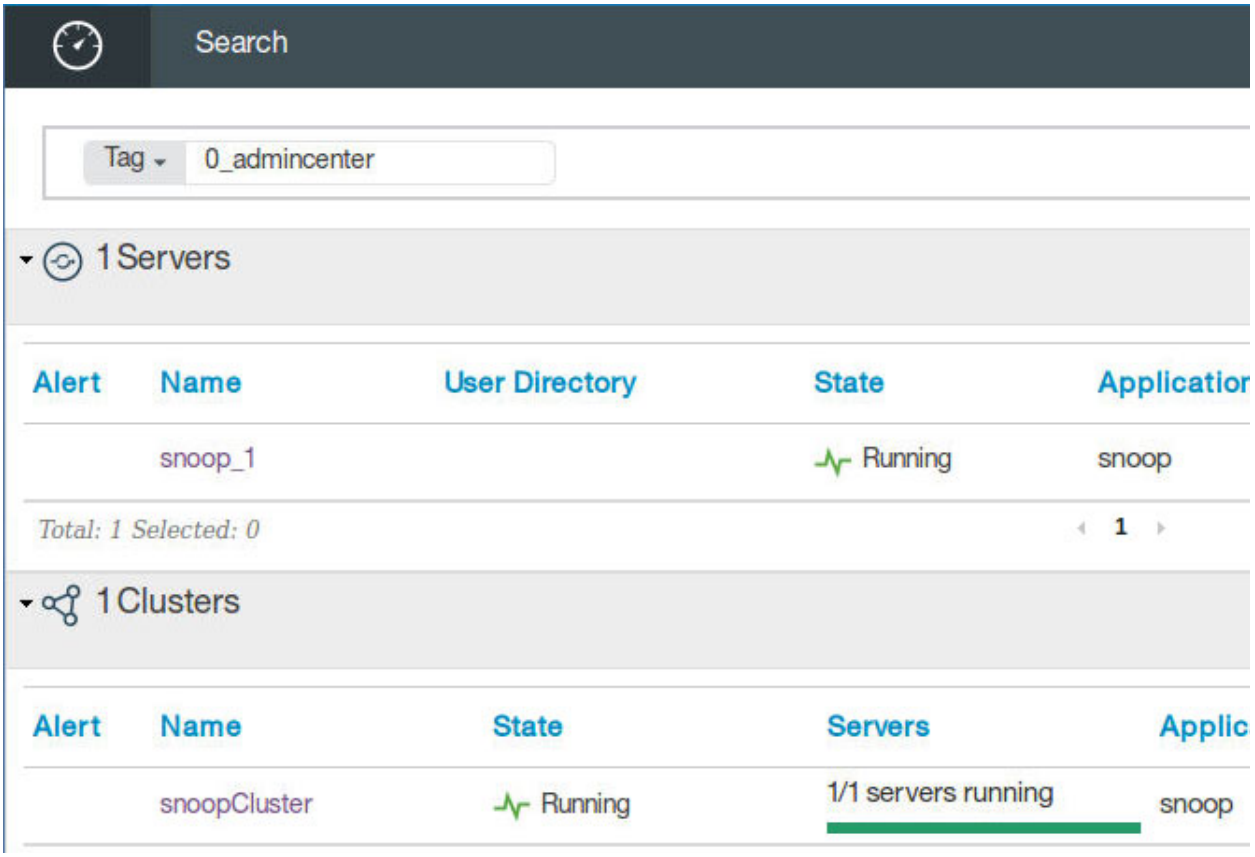
__ m. In the search field, click the **x** icon to remove the existing search parameters.

__ 2. Create another search.

__ a. In the Search field, click the **+** icon and change the search criterion to **Tag**.

__ b. For a value, enter: `0_admincenter`

__ c. Click the **Search** icon to begin the search. The results indicate that both a server and a cluster are tagged with `0_admincenter` metadata.



__ d. Feel free to experiment with the search tool and think about how it might help you quickly find the resources that you need. The search tool is useful if you have many resources that are being managed.

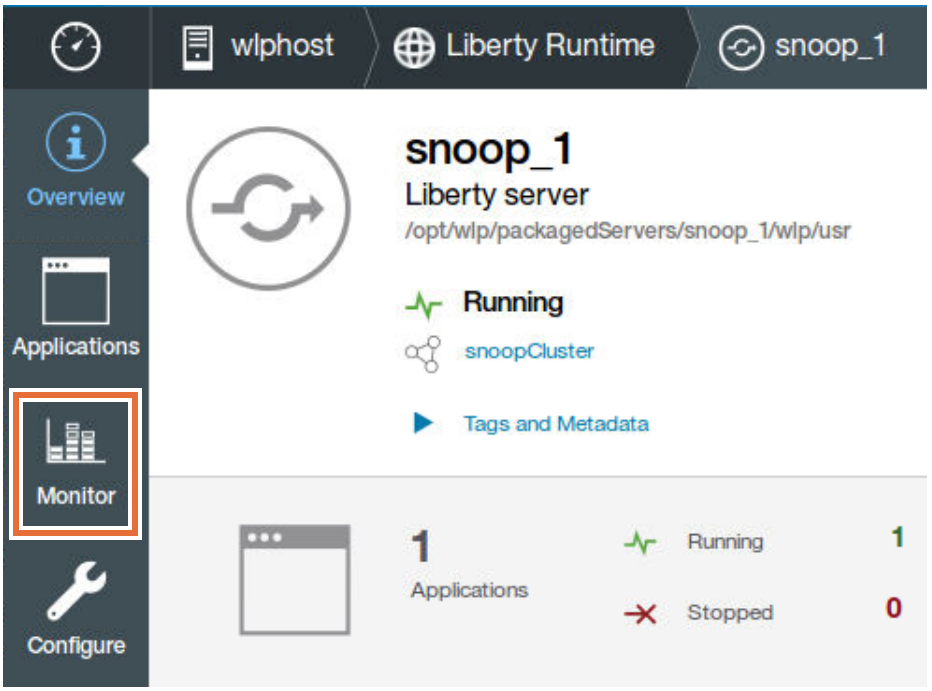
__ 3. View performance metrics in the Admin Center.

__ a. Click the **Dashboard** icon to go back to the dashboard.

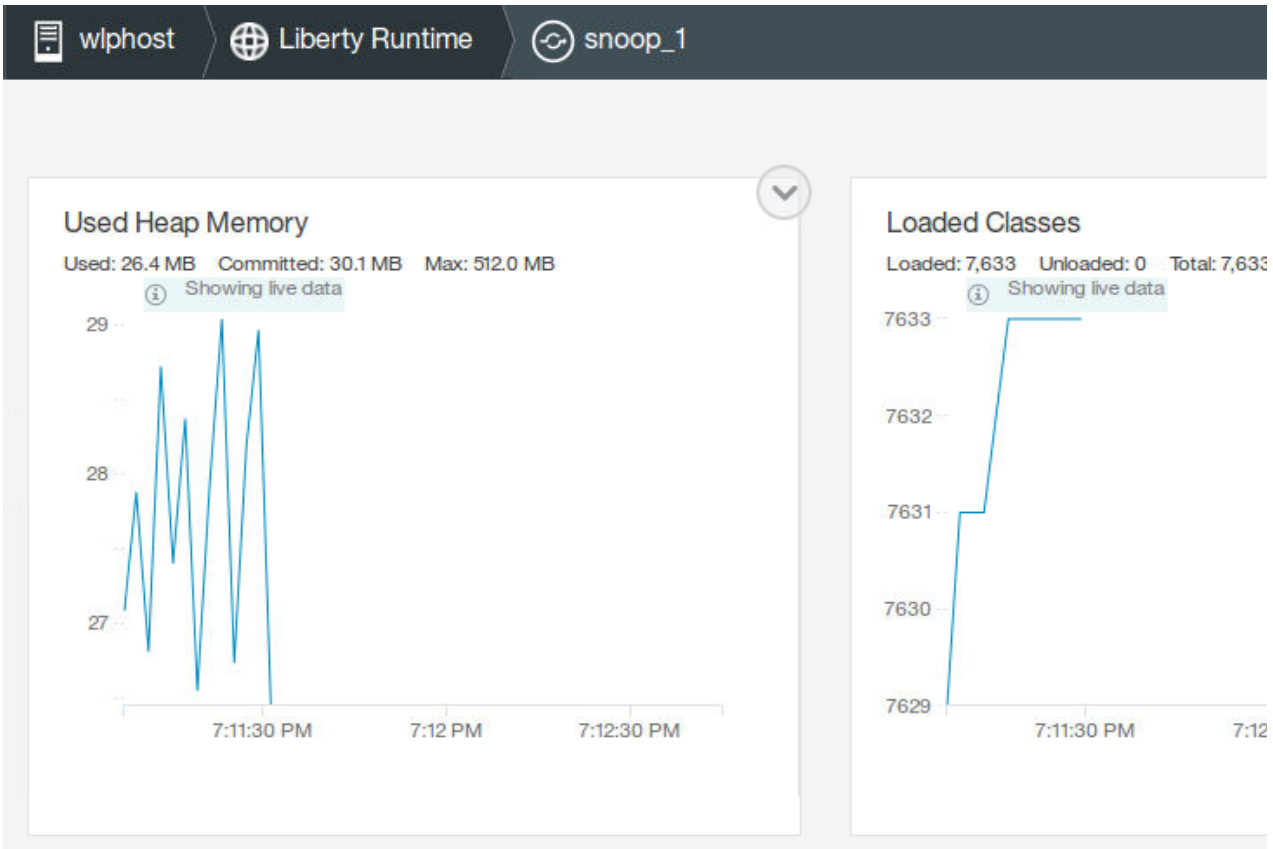
__ b. Go to the SERVERS section, and click the **SERVERS** pane.

__ c. Click the icon for the **snoop_1** server.

__ d. In the navigation on the left, click **Monitor**.



__ e. The basic statistics for heap, loaded classes, threads, and CPU are available and are updated.



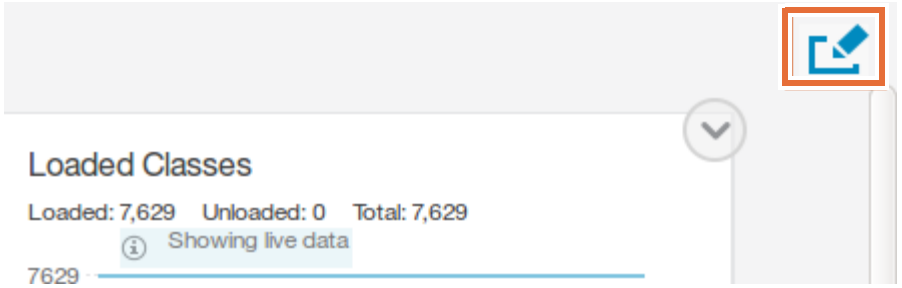
Information

When first displayed, the Monitor view shows the following charts:

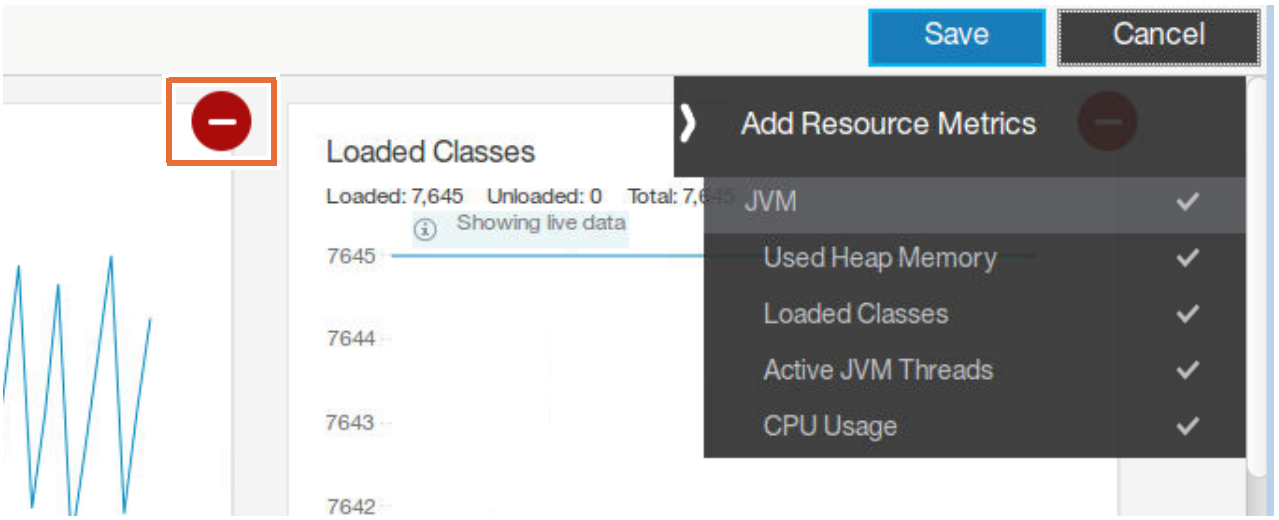
- **Used Heap Memory**: This chart shows the heap memory that is used by the server in megabytes (MB) every 2 seconds. The chart also shows the used, committed, and maximum megabytes.
- **Loaded Classes**: This chart shows the number of classes that are loaded every 2 seconds. The chart also shows the number of loaded and unloaded classes, plus the total number of classes.
- **Active JVM Threads**: This chart shows the number of JVM threads every 2 seconds. The chart also shows the number of live, total, and peak threads.
- **CPU Usage**: This chart shows the percentage of CPU that is used every 2 seconds.

If a server has the `monitor-1.0` feature enabled, the Monitor view also has other charts, depending on the resource.

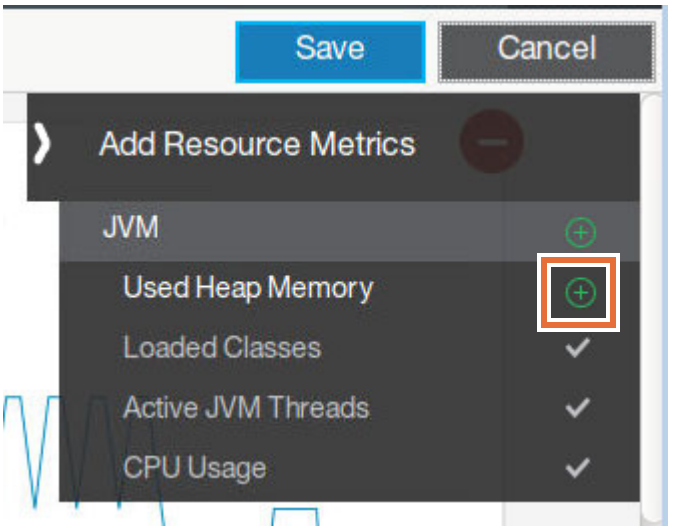
__ f. You can select to show or hide charts on the Monitor view. To control which metrics to display, click the **Edit Charts** icon.



__ g. Each resource metric on the list shows a check mark next to the name. This check mark indicates the metric is displayed. To remove a metric that you do not want to display, click the red **-** icon on the metric. On the Used Heap Memory graphic, click the red **-** icon.



__ h. You can see that the metric is removed from the page and the menu changes. There is now a green + next to Used Heap Member. Click the green **+** to add a metric that you want to display. Click the green **+** next to Used Heap Memory to add it back to the page.



__ i. When your changes are completed, click **Save**.

Part 7: Cleaning up the environment

In this part of the exercise, you clean up the deployed artifacts.

__ 1. Stop the snoop_1 server by using the Admin Center.

__ a. Click the **Dashboard** icon to go back to the dashboard.

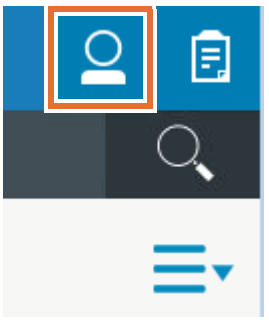
__ b. Go to the SERVERS section, and click the **SERVERS** pane.

__ c. Click the **Actions** menu icon for snoop_1, and click **Stop**.

__ d. In the Stop server snoop_1 window, click **Stop**.

__ 2. Log out of the Admin Center.

__ a. To log out of the Admin Center, click the **User** icon.



__ b. From the menu, click **Log out admin**.

__ c. Close the browser window.

__ 3. The Admin Center does not yet support deletion of your packaged server. However, you can complete this task by using the command line. First, remove the collective member, snoop_1.

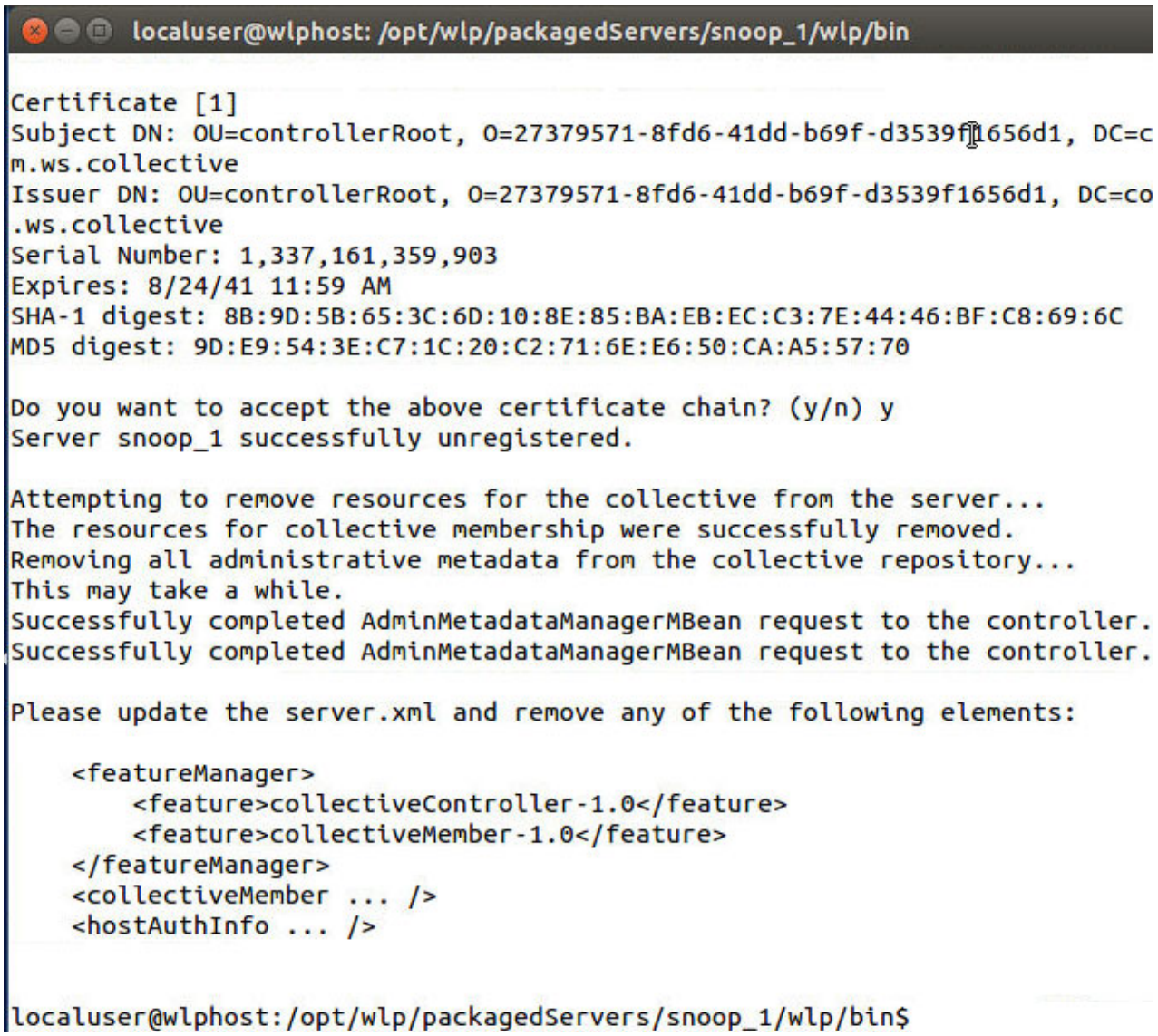
__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/packagedServers/snoop_1/wlp/bin` directory.

__ c. To remove the server, enter the following command:

```
./collective remove snoop_1 --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the certificate chain.



__ d. Go to the `/opt/wlp/packagedservers` directory.

__ e. To remove the snoop_1 directory, enter the following command:

```
rm -rf snoop_1
```

The `packagedservers` directory is now empty.

__ 4. Stop the server adminCenterController.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop adminCenterController
```

__ c. Exit the terminal window.

End of exercise

Exercise review and wrap-up

In this exercise, you created a server and configured it to be a collective controller. Then, a packaged server is created and deployed to the Liberty run time by using the Admin Center. Finally, you examine some of the features of the Admin Center, which included the search tool and monitoring metrics.

Exercise 2. WebSphere Liberty administration by using Jython Scripts

Estimated time

01:30

Overview

In this exercise, you use command line and Jython scripting tools to deploy and maintain clustered application servers centrally through the collective controller.

Objectives

After completing this exercise, you should be able to:

- Deploy an application to a vertical cluster of packaged Liberty servers
- Use Jython scripts to get the status of a cluster
- Use Jython scripts to start and stop servers and clusters
- Generate a web server plug-in
- Access an application through an HTTP server
- Review whether HTTP failover occurs through the HTTP server

Introduction

Liberty supports scripting in any language though it is recommended that you use a Java scripting language such as Jython, JRuby, Groovy, and more. Liberty includes a Jython library, `restConnector.py`, which provides a Jython interface to the Representational State Transfer (REST) Java Management Extensions (JMX) connector of Liberty. This connector is tested with Jython and is compatible with versions 2.5.4 and higher. Liberty does not include with a Jython run time so you need to download and configure the run time. The Jython binary files are downloaded and placed on the course image for use in this exercise.

Requirements

To complete this exercise, you need the WebSphere Liberty binary files and Liberty installed. You also need the Jython binary files.

Exercise instructions

Part 1: Installing Jython

You can use Jython-based scripts to establish a Java Management Extensions (JMX) MBean Liberty server connection. In this part of the exercise, you install the Jython computer language.

__ 1. Install Jython.

__ a. Open a terminal window and go to the `/opt/software/WLP_16.0.0.2` directory.

__ b. To install Jython v2.5.3, enter the following command:

```
java -jar jython-installer-2.5.3.jar
```

Hint

If you get an error, verify that the JAVA_HOME environment variable is set. To verify, enter the following command:

```
java -version
```

If you do not get the details on the Java version, enter the following command:

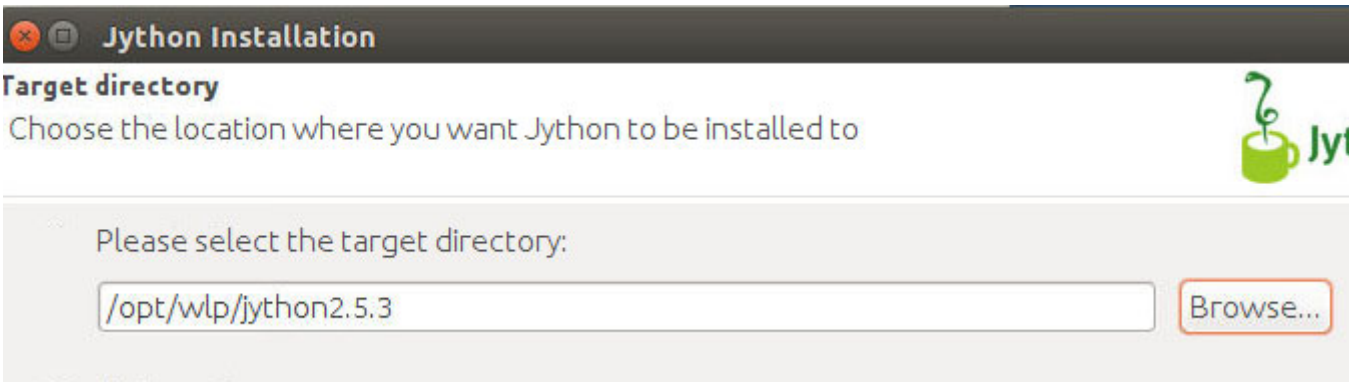
```
source /etc/environment
```

__ c. On the Welcome pane, keep the default language selected and click **Next**.

__ d. On the License agreement pane, select the option **I accept** and click **Next**.

__ e. On the Installation type pane, verify that the option **Standard (core, library modules, demos, examples, documentation)** is selected and click **Next**.

__ f. In the target directory field, enter: `/opt/wlp/jython2.5.3`



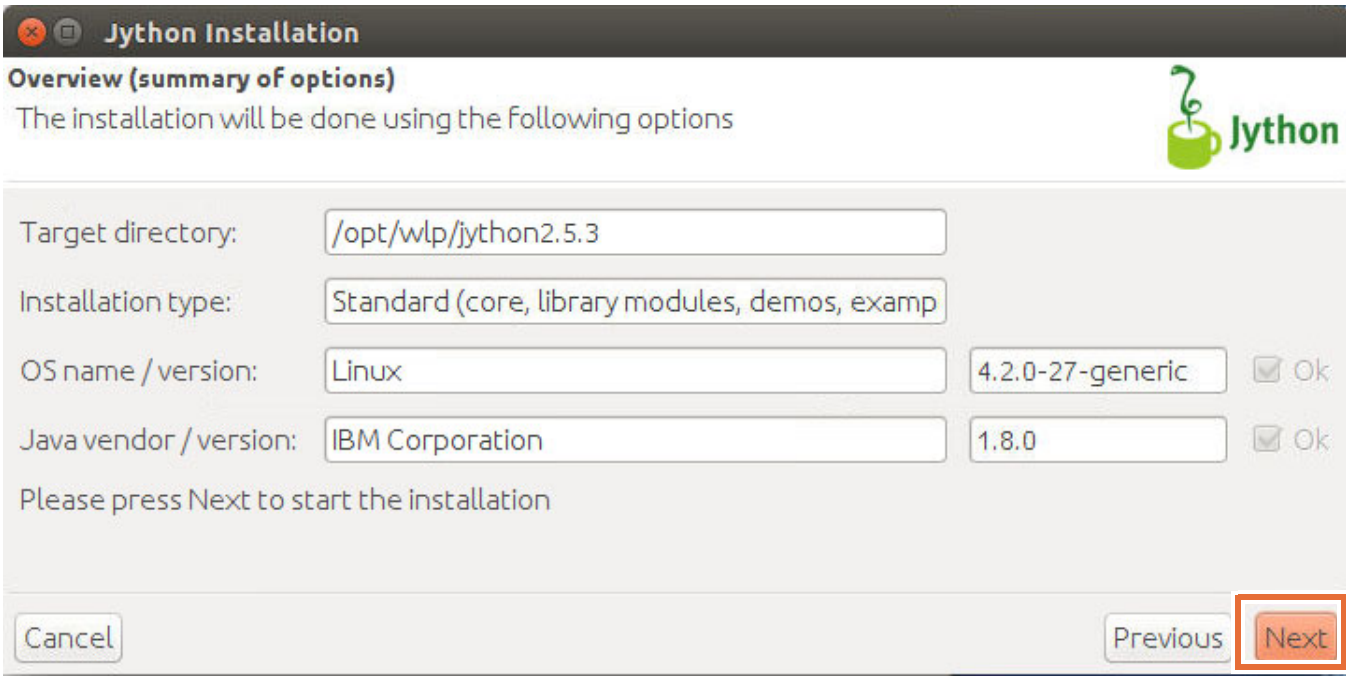
__ g. Click **Next**.

__ h. On the Information window that indicates the directory is created, click **OK**.

__ i. Click **Next**.

__ j. On the Target java home pane, verify that **Current** is selected and click **Next**.

__ k. Review the summary information. When completed, click **Next**.



__ l. Wait for the installation to complete. On the Readme pane, click **Next**.

__ m. On the final pane, click **Finish**.

__ 2. Update the required environment variables.

__ a. To use Jython, you need to add it to the system PATH environment variable. To add Jython to the PATH environment variable, enter the following command:

```
export PATH=$PATH:/opt/wlp/jython2.5.3/bin
```

__ b. To verify the environment variable setting, enter the following command:

```
echo $PATH
```

__ c. To run the sample scripts, you need to set the CLASSPATH and JYTHONPATH environment variables. Jython uses these variables to locate the required libraries, such as the restConnector.jar and sample script libraries. To set the CLASSPATH environment variable, enter the following command:

```
export CLASSPATH=/opt/wlp/clients/restConnector.jar
```

__ d. To set the JYTHONPATH environment variable, enter the following command:

```
export JYTHONPATH=/opt/wlp/clients/jython/:/opt/labfiles/management/lib/
```

Part 2: Deploying the packaged server by using scripting

In this section, you use sample Jython scripts that perform the following tasks:

- Registers the host. If the host is already registered, it updates with any new information you provide.
- Uploads the packaged server and extracts it into the destination directory.
- Set up security certificates.
- Joins the members with the collective.
- Starts and stops the application server member1 and member2.

__ 1. Examine the management labfiles.

__ a. Go to the `/opt/labfiles/management/` directory.

__ b. List the contents of the directory. You can see various Jython scripts, which end in `.py`. You use these scripts in the exercise to complete various functions by using scripting, instead of the Admin Center GUI. Feel free to open each of the script files with an editor and examine the contents of the files.

__ c. Go to the `/opt/labfiles/management/snoop2` directory.

__ d. List the contents of the directory. You can see the file `snoop2.zip`. The file contains a packaged server. It is a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server.

__ e. Go to the `/opt/labfiles/management/snoop2/wlp/usr/servers` directory.

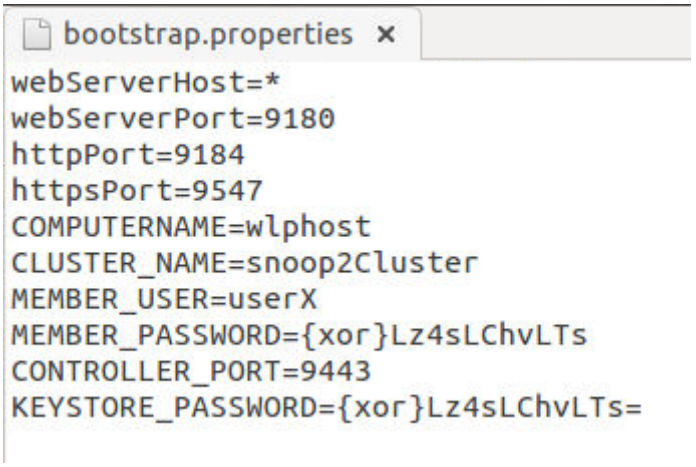
__ f. List the contents of the directory. You can see two servers, snoop2_a and snoop2_b.

__ g. Go to the `snoop2_a` directory and list the contents.

```
localuser@wlphost: /opt/labfiles/management/snoop2/wlp/usr/servers/snoop2_a
localuser@wlphost:/opt/labfiles/management/snoop2/wlp/usr/servers/snoop2_a$ ls
admin-metadata.xml  bootstrap.properties        server.xml
apps                collective-join-include.xml
localuser@wlphost:/opt/labfiles/management/snoop2/wlp/usr/servers/snoop2_a$
```

You can see various files and a subdirectory that contains an application. Both snoop2_a and snoop2_b have the same files.

__ h. Open the `bootstrap.properties` file and examine the default settings. Open the file by using an editor such as vi or gedit.



Notice the details and the **httpPort** number, which is 9184.

__ i. Close the file when completed.

__ j. Feel free to open each of the remaining files with an editor and examine the contents of the files.

__ k. Go to the `/opt/labfiles/management/snoop2/wlp/usr/servers/snoop2_b` directory.

Open the `bootstrap.properties` file and examine the default settings. Open the file by using an editor such as vi or gedit. Notice the **httpPort** number, which is 9185.

__ l. Close the file when completed.

__ m. Feel free to open each of the remaining files with an editor and examine the contents of the files.

Information

If you examine each of the files, you can see the following details:

- Each `server.xml` file includes a copy `collective-join-include.xml` required for registration with controller.
- Each server is a member of a cluster that is called `snoop2Cluster` that is defined in the `bootstrap.properties` file.
- Each directory includes the `apps` subdirectory, which contains a copy of the application `snoop2.war`.

__ 2. Start the adminCenterController server.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To start the server, enter the following command:

```
./server start adminCenterController
```

__ 3. Deploy cluster members by using the `deployMembers.py` script.

__ a. Go to `/opt/wlp/jython2.5.3` directory.

Note

When entering a command with various parameters, it does not matter what order you place the parameters. It is a requirement that you use the proper syntax and parameter name, but the order does not matter.

__ b. To deploy cluster members, use the `deployMembers.py` script by entering the following command:

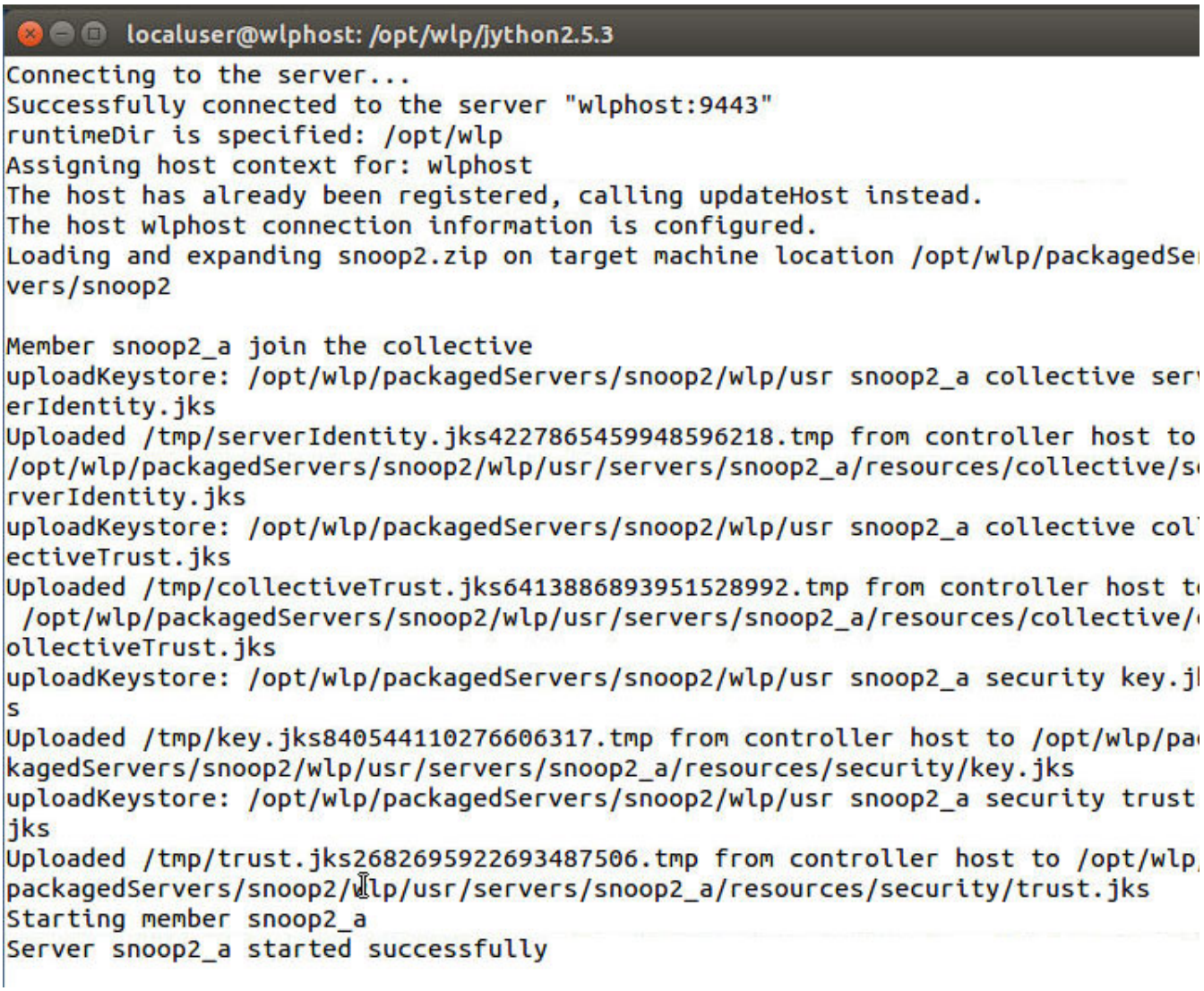
```
./jython /opt/labfiles/management/deployMembers.py
--zipFile=/opt/labfiles/management/snoop2/snoop2.zip
--installDir=/opt/wlp/packagedServers/snoop2 --runtimeDir=/opt/wlp
--installHost=wlphost --rpcUser=localuser --rpcUserPassword=passw0rd
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

Information

The command that is entered includes the following details:

- You supply the image user ID and password by using the `--rpcUser` and `--rpcUserPassword` parameters to allow the servers to be started and stopped remotely through the controller.
- You install the servers to `/opt/wlp/packagedServers/snoop2` directory rather than directly under an existing Liberty installation. This capability gives you flexibility to separate the location of the run time and the servers.
- The `--runtimeDir` parameter specifies the directory for the Liberty run time. This information is stored with the controller so that it is able to use the correct run time when starting or stopping the members.

__ c. Examine the output from the command. You can see that the Jython packages are loaded the first time that you enter a Jython command. The `deployMembers.py` script connects to the server, calls `updateHost`, then loads, and expands the packaged server to the target location.



The members, snoop2_a and snoop2_b, are added to the collective and each server is started. You should see the messages that the server snoop2_a and snoop2_b started successfully.

__ 4. Examine the file structure after running the script.

__ a. Go to the `/opt/wlp/packagedServers/snoop2/wlp/usr/servers/snoop2_a/logs` directory.

__ b. List the contents of the directory. You can see log files and a state subdirectory.

__ c. Open the `messages.log` file and verify that there are no errors. Open the file by using an editor such as vi or gedit.

__ d. When completed, close the file.

__ e. Repeat the steps for snoop2_b. Go to the `/opt/wlp/packagedServers/snoop2/wlp/usr/servers/snoop2_b/logs` directory.

__ f. Open the `messages.log` file and verify that there are no errors. Open the file by using an editor such as vi or gedit.

__ g. When completed, close the file.

Note

If you get the following error:

```
ImportError: No module named restConnector
```

Use the `echo` command to check that you entered the export commands properly when setting your environment variables.

Part 3: Working with more administrative Jython scripts

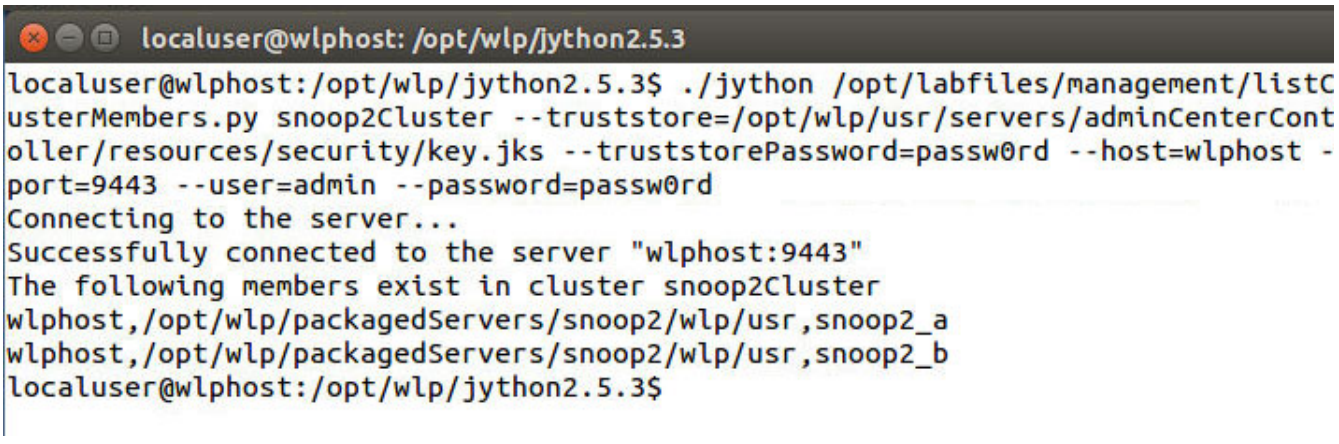
In this section, you use some sample administrative Jython scripts to list the cluster members, stop and start the cluster, and start and stop cluster members.

__ 1. List the cluster members.

__ a. Go to `/opt/wlp/jython2.5.3` directory.

__ b. List the collective member by using the `snoop2Cluster.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/listClusterMembers.py snoop2Cluster
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/ke
y.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

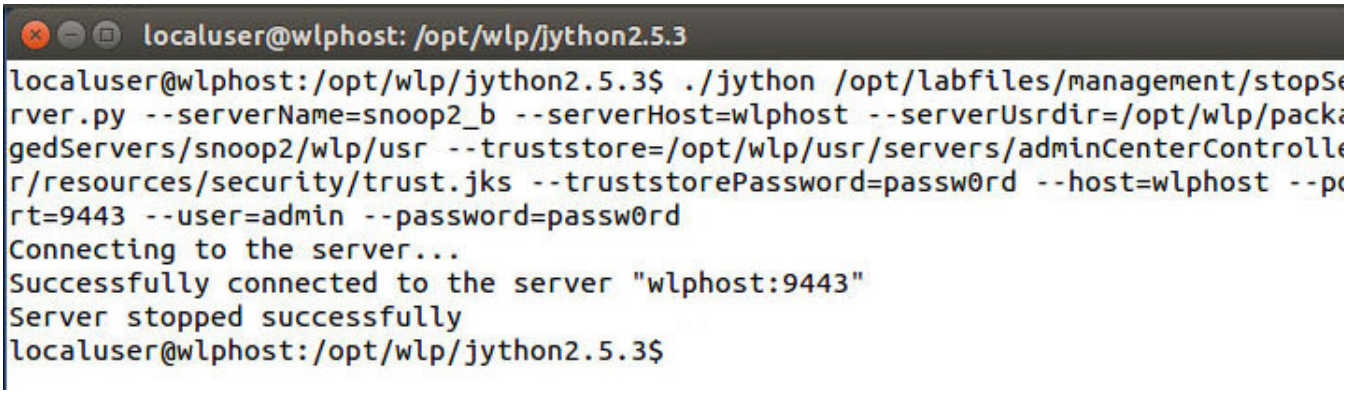


Verify that both snoop2_a and snoop2_b are listed.

__ 2. Start and stop the cluster members.

__ a. You can stop the cluster members by using the `stopServer.py` script. In this example, stop snoop2_b. To use the script to stop snoop2_b, enter the following command:

```
./jython /opt/labfiles/management/stopServer.py --serverName=snoop2_b
--serverHost=wlphost --serverUsrdir=/opt/wlp/packagedServers/snoop2/wlp/usr
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```



__ b. Verify that the cluster status is PARTIALLY STARTED, which means not all cluster members are started. To verify the status, use the `getClusterStatus.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/getClusterStatus.py snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/ke
y.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

You can see that the status for snoop2Cluster is PARTIALLY STARTED.

__ c. Start the collective member, snoop2_b by using the `startServer.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/startServer.py --serverName=snoop2_b
--serverHost=wlphost --serverUsrdir=/opt/wlp/packagedServers/snoop2/wlp/usr
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

```
localuser@wlphost: /opt/wlp/jython2.5.3
localuser@wlphost:/opt/wlp/jython2.5.3$ ./jython /opt/labfiles/management/startS
erver.py --serverName=snoop2_b --serverHost=wlphost --serverUsrdir=/opt/wlp/pack
agedServers/snoop2/wlp/usr --truststore=/opt/wlp/usr/servers/adminCenterControll
er/resources/security/trust.jks --truststorePassword=passw0rd --host=wlphost --p
ort=9443 --user=admin --password=passw0rd
Connecting to the server...
Successfully connected to the server "wlphost:9443"
Server started successfully
localuser@wlphost:/opt/wlp/jython2.5.3$
```

__ d. Verify that the cluster status as STARTED, which means all cluster members are started. Again, use the `getClusterStatus.py` script to verify the status. To use the script, enter the following command:

```
./jython /opt/labfiles/management/getClusterStatus.py snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/ke
y.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

```
localuser@wlphost: /opt/wlp/jython2.5.3
localuser@wlphost:/opt/wlp/jython2.5.3$ ./jython /opt/labfiles/management/getCl
sterStatus.py snoop2Cluster --host=wlphost --port=9443 --truststore=/opt/wlp/us
/servers/adminCenterController/resources/security/key.jks --truststorePassword=
assw0rd --user=admin --password=passw0rd
Connecting to the server...
Successfully connected to the server "wlphost:9443"
Status for cluster snoop2Cluster: STARTED
localuser@wlphost:/opt/wlp/jython2.5.3$
```

You now have experience using the sample administrative Jython scripts on the collective controller to complete operations against collective members.

__ 3. Stop and start the clusters.

__ a. Stop the cluster by using the `stopCluster.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/stopCluster.py --clusterName=snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

You can see that both snoop2_a and snoop2_b are both stopped.

__ b. Start the cluster by using the `startCluster.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/startCluster.py --clusterName=snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

You can see that both snoop2_a and snoop2_b are both started.

Part 4: Generating the web server plug-in by using Jython

A web server plug-in is used to forward HTTP requests from a supported web server to one or more Liberty servers. The plug-in takes a request and checks the request against configuration data in the `plugin-cfg.xml` file. The configuration data maps the URI for the HTTP request to the host name of the Liberty server. The web server plug-in then uses this information to forward the request to the Liberty server.

In this part, you generate a custom `plugin-cfg.xml` file that is named `snoop2Cluster-plugin-cfg.xml`.

__ 1. Verify that all the Liberty servers are running and able to process requests by accessing the SnoopServlet.

__ a. Open the Firefox web browser to member1, snoop2_a. Use the **httpPort** number that you noted earlier from the `bootstrap.properties` file. Enter the following URL:

`http://wlphost:9184/snoop2/SnoopServlet`

__ b. Go to member2, snoop2_b. Use the **httpPort** number that you noted earlier from the `bootstrap.properties` file. Enter the following URL:

`http://wlphost:9185/snoop2/SnoopServlet`

__ c. Minimize the web browser.

__ 2. Generate the `plugin-cfg.xml` file for your Liberty servers and applications.

__ a. Go to the terminal window.

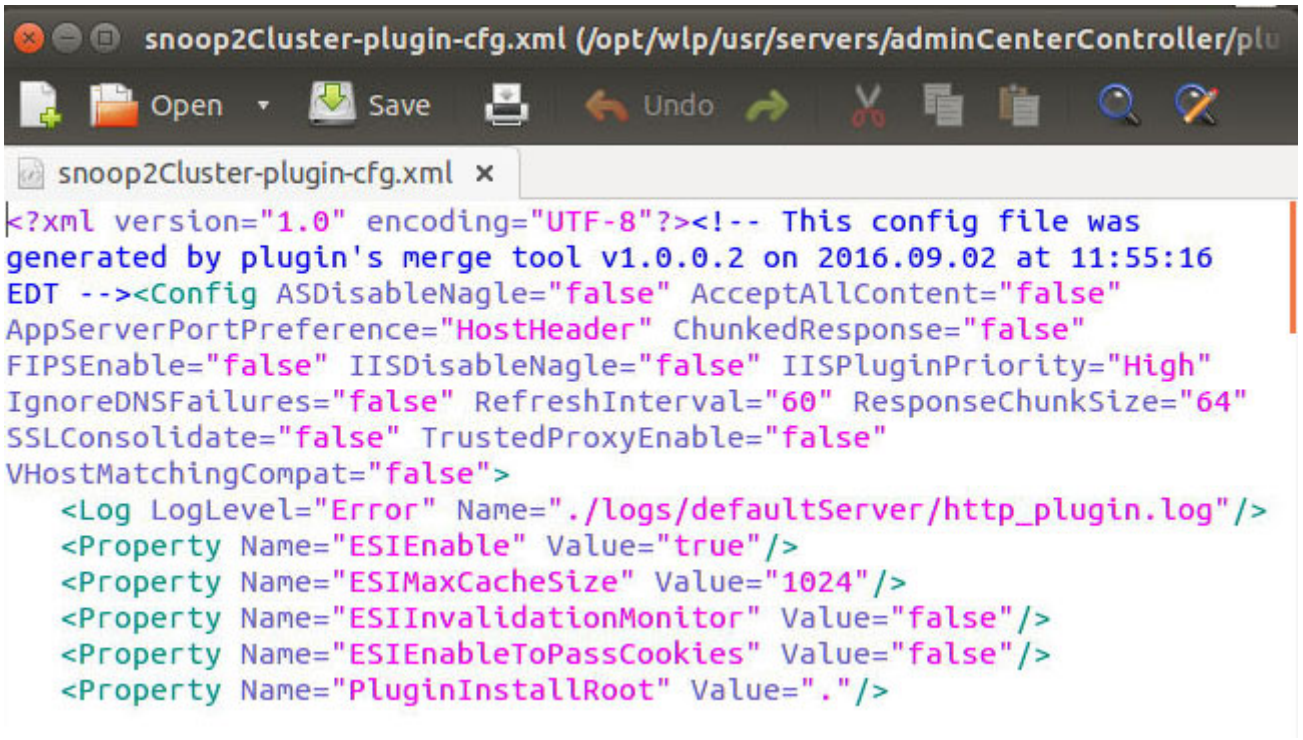
__ b. To generate the `plugin-cfg.xml` file, use the `genClusterPlugin.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/genClusterPlugin.py snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

__ 3. Examine the generated `snoop2Cluster-plugin-cfg.xml` file.

__ a. Go to the `/opt/wlp/usr/servers/adminCenterController/pluginConfig` directory.

__ b. Open the `snoop2Cluster-plugin-cfg.xml` file by using an editor, such as vi or gedit.



__ c. Examine the details in the file. Go to the Virtual Hosts Groups section. You can see that it is suitable for use by a web server that is configured with host “” and port 9180.

__ d. When completed, close the file.

Part 5: Testing the web server plug-in with IBM HTTP Server

The IBM HTTP Server and the web server plug-in are already installed on the course image.

In this section, you configure the web server plug-in so that, when the web server receives an HTTP request for dynamic resources, the request is forwarded to the Liberty controller.

__ 1. Configure the IBM HTTP Server and plug-in.

__ a. Go to the HTTP Server `/opt/IBM/HTTPServer/conf` configuration directory.

__ b. Make a backup of the `httpd.conf` file. To make a backup, enter the following command:

```
cp httpd.conf httpd-backup.conf
```

__ c. Open the `httpd.conf` file and examine the default settings. Open the file by using an editor such as vi or gedit.

__ d. In the `httpd.conf` file, make the following changes:

- Change `Listen 8080` to `Listen 9180`
- Change the last line in the file to:
  `WebSpherePluginConfig`
  `/opt/IBM/WebSphere/Plugins/config/IHS1/snoop2Cluster-plugin-cfg.xml`

__ e. Save and close the file when completed.

__ f. Go to the `/opt/wlp/usr/servers/adminCenterController/pluginConfig` directory.

__ g. Copy the `snoop2Cluster-plugin-cfg.xml` file to the `/opt/IBM/WebSphere/Plugins/config/IHS1/` directory. To copy the file, enter the following command:

```
cp snoop2Cluster-plugin-cfg.xml /opt/IBM/WebSphere/Plugins/config/IHS1/
```

__ h. Go to the `/opt/IBM/WebSphere/Plugins/config` directory.

__ i. Create a directory named `defaultServer` to accommodate the name of the log file for the plug-in. To create the directory, enter the following command:

```
mkdir defaultServer
```

This directory is defined in the `snoop2Cluster-plugin-cfg.xml` file as `<Log LogLevel="Error" Name=".\logs\defaultServer\http_plugin.log"/>`.

Note

You can also make each of these changes by using the `setSnoopClusterPlugin.sh` script.

__ 2. Start the IBM HTTP Server

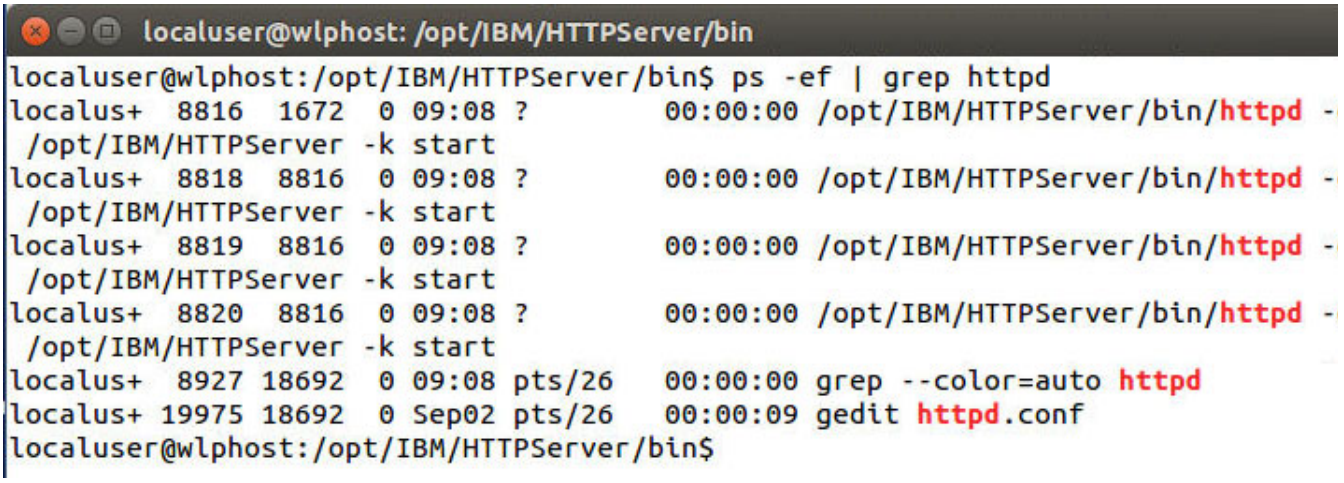
__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. To start the HTTP Server, enter the following command:

```
./apachectl start
```

__ c. To see whether the HTTP Server is started, enter the following command:

```
ps -ef | grep httpd
```



You can see that a number of processes are started.

__ 3. Test the application.

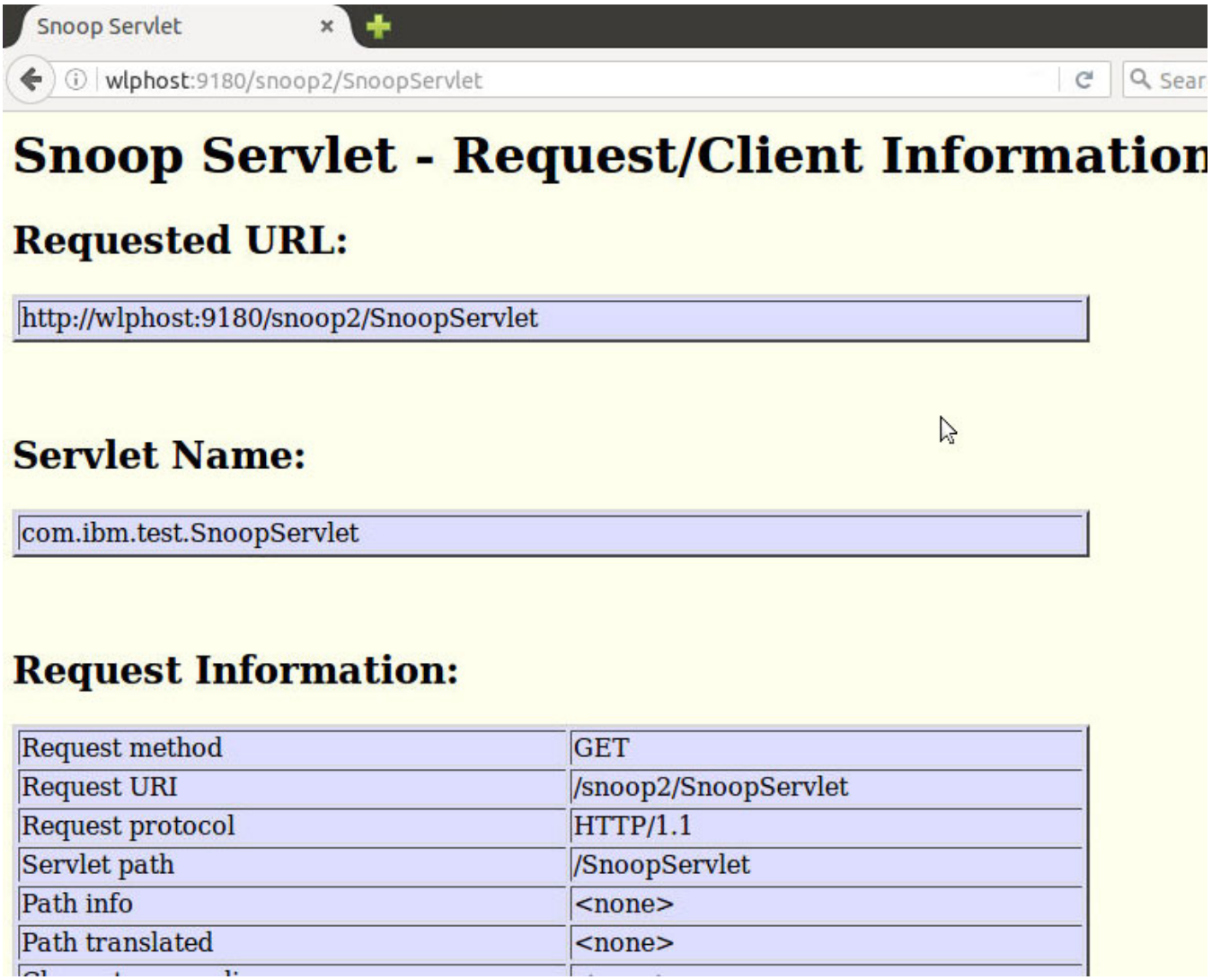
__ a. Open a Firefox web browser and direct it to the web server by using the port 9180. This is the port that you entered in the `httpd.conf` file. To test, enter the following URL:

`http://wlphost:9180`

You can see the main page for the IBM HTTP Server.

__ b. Test that the web server is routing requests to the snoop2 cluster by entering the following URL:

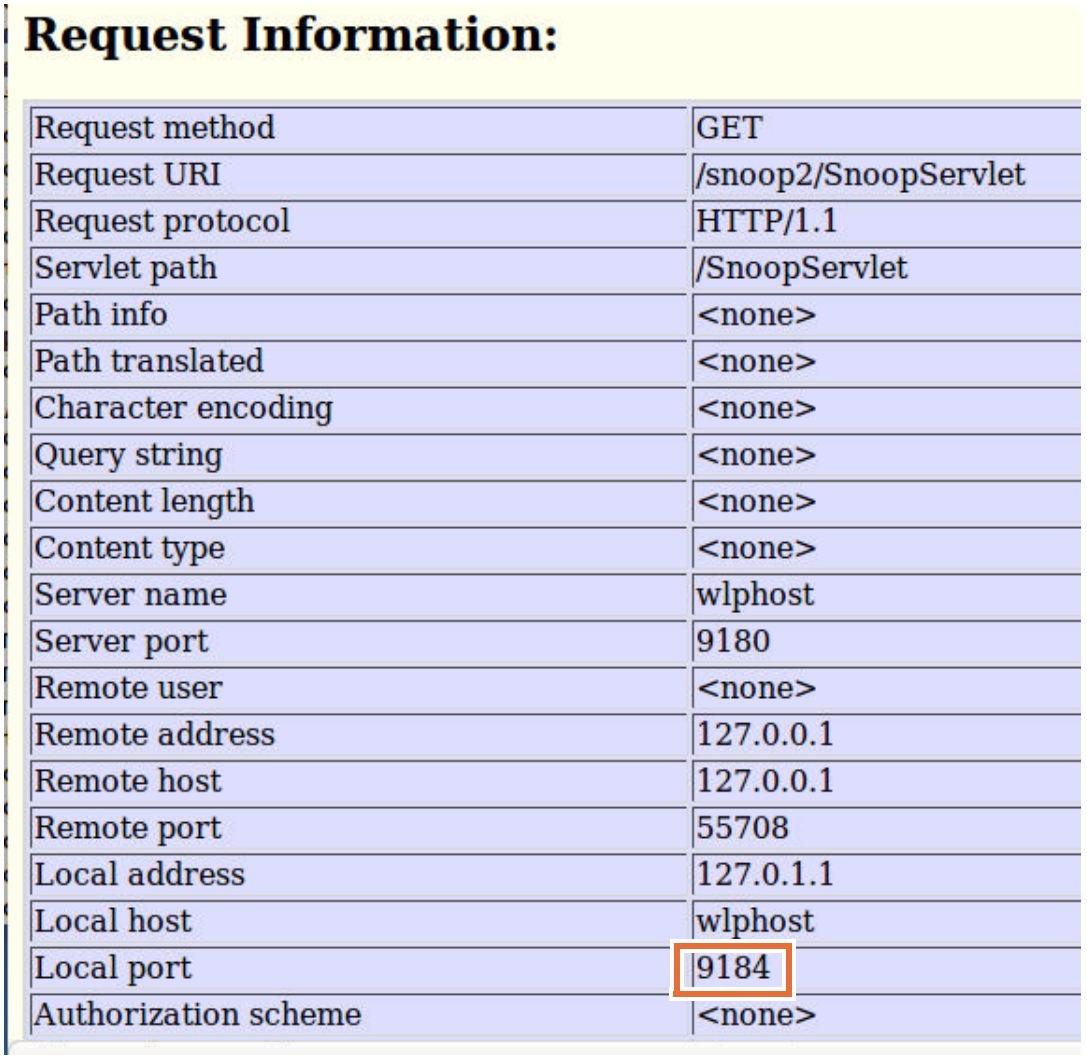
`http://wlphost:9180/snoop2/SnoopServlet`



__ c. In the Request Information section, examine the **Local port** information.

__ d. Check for the Local port number, which should be the listener port of either snoop2_a or snoop2_b. Look for the following details:

- Port 9184 is for snoop2_a
- Port 9185 is for snoop2_b

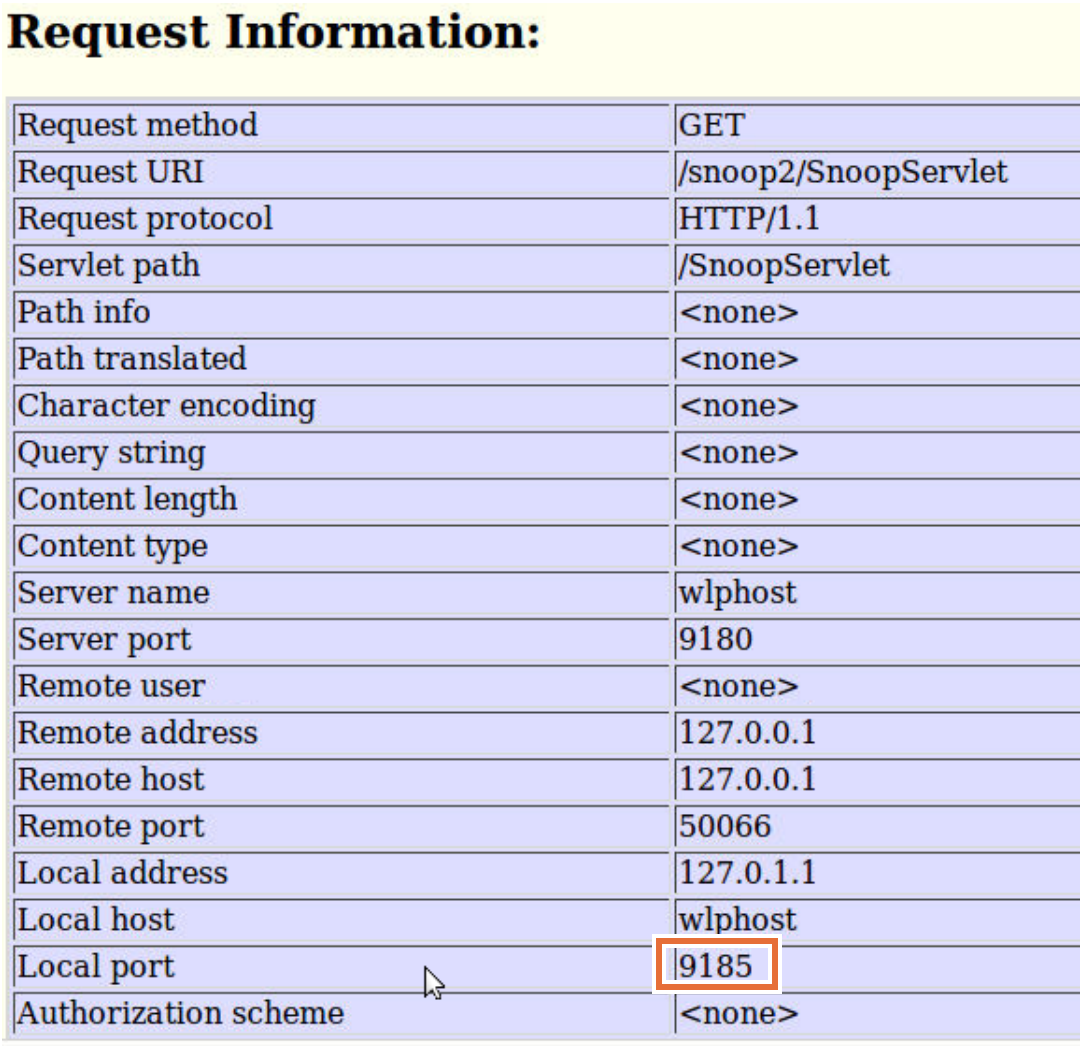


Request Information:

Request method	GET
Request URI	/snoop2/SnoopServlet
Request protocol	HTTP/1.1
Servlet path	/SnoopServlet
Path info	<none>
Path translated	<none>
Character encoding	<none>
Query string	<none>
Content length	<none>
Content type	<none>
Server name	wlphost
Server port	9180
Remote user	<none>
Remote address	127.0.0.1
Remote host	127.0.0.1
Remote port	55708
Local address	127.0.1.1
Local host	wlphost
Local port	9184
Authorization scheme	<none>

Recall these are the ports that are defined in the `bootstrap.properties` file for each server.

__ e. Refresh the browser a few times, and check that the web server round robins between the two servers.



Request Information:

Request method	GET
Request URI	/snoop2/SnoopServlet
Request protocol	HTTP/1.1
Servlet path	/SnoopServlet
Path info	<none>
Path translated	<none>
Character encoding	<none>
Query string	<none>
Content length	<none>
Content type	<none>
Server name	wlphost
Server port	9180
Remote user	<none>
Remote address	127.0.0.1
Remote host	127.0.0.1
Remote port	50066
Local address	127.0.1.1
Local host	wlphost
Local port	9185
Authorization scheme	<none>

__ f. Minimize the browser window.

__ 4. Stop and start the collective members.

__ a. Go to `/opt/wlp/jython2.5.3` directory.

__ b. Stop the collective member that processed your request by using the `stopServer.py` script. To use the script to stop snoop2_b, enter the following command:

```
./jython /opt/labfiles/management/stopServer.py --serverName=snoop2_b
--serverHost=wlphost --serverUsrdir=/opt/wlp/packagedServers/snoop2/wlp/usr
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

__ c. Refresh the browser. The new page should display the port number for the remaining server.

__ d. Stop the remaining server by using the `stopServer.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/stopServer.py --serverName=snoop2_a
--serverHost=wlphost --serverUsrdir=/opt/wlp/packagedServers/snoop2/wlp/usr
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

__ e. Refresh the browser. You get an internal server error.

__ f. Start both servers, or start the cluster, by using the `startCluster.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/startCluster.py --clusterName=snoop2Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

__ g. Refresh the browser until the page appears again, which might take a minute. You see that the web server round robins between the two hosts again.

__ h. Minimize the browser window.

Part 6: Application updates without downtime

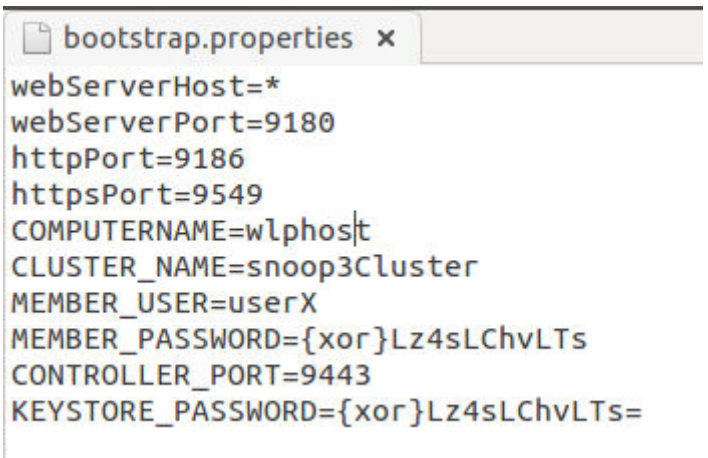
In this part of the exercise, you deploy a new version of the snoop2 application called snoop3. You use the load balancing weight of the plug-in configuration to seamlessly transition from the old version to the new version. This is one manual way to achieve continuous application availability. With this approach, you can keep as many versions as you want, and you are also able to roll back to previous versions.

__ 1. Review the snoop3 directory structure.

__ a. In the terminal window, go to the `/opt/labfiles/management/snoop3/wlp/usr/servers/snoop3_a` directory.

__ b. List the contents of the directory.

__ c. Open the `bootstrap.properties` file and examine the default settings. Open the file by using an editor such as vi or gedit.



Notice the details and the **httpPort** number, which is 9186.

__ d. Close the file when completed.

__ e. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

server.xml

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
        host="${COMPUTERNAME}"
        httpPort="${httpPort}"
        httpsPort="${httpsPort}" />

    <virtualHost id="default_host"
allowFromEndpointRef="defaultHttpEndpoint" >
        <hostAlias>${COMPUTERNAME}:${httpPort}</hostAlias>
        <hostAlias>${COMPUTERNAME}:${httpsPort}</hostAlias>
        <hostAlias>${webServerHost}:${webServerPort}</hostAlias>
    </virtualHost>

    <webApplication id="snoop3" location="snoop3.war"
contextRoot="snoop2" />
```

The file is basically the same as snoop2 except for the following differences:

- The new application is called snoop3
- The application has the same context root, snoop2. This feature allows you to configure the plug-in to route to both applications snoop2 and snoop3 as a single HTTP cluster. But, you are still managing them as separate applications.
- The cluster name is snoop3Cluste and the server names are snoop3_a and snoop3_b.

__ f. Close the file when completed.

__ g. Feel free to open each of the remaining files with an editor and examine the contents of the files.

__ h. Go to the `/opt/labfiles/management/snoop3/wlp/usr/servers/snoop3_b` directory.

Open the `bootstrap.properties` file and examine the default settings. Open the file by using an editor such as vi or gedit. Notice the **httpPort** number, which is 9187.

__ i. Close the file when completed.

__ j. Feel free to open each of the remaining files with an editor and examine the contents of the files.

__ 2. Deploy the snoop3 package.

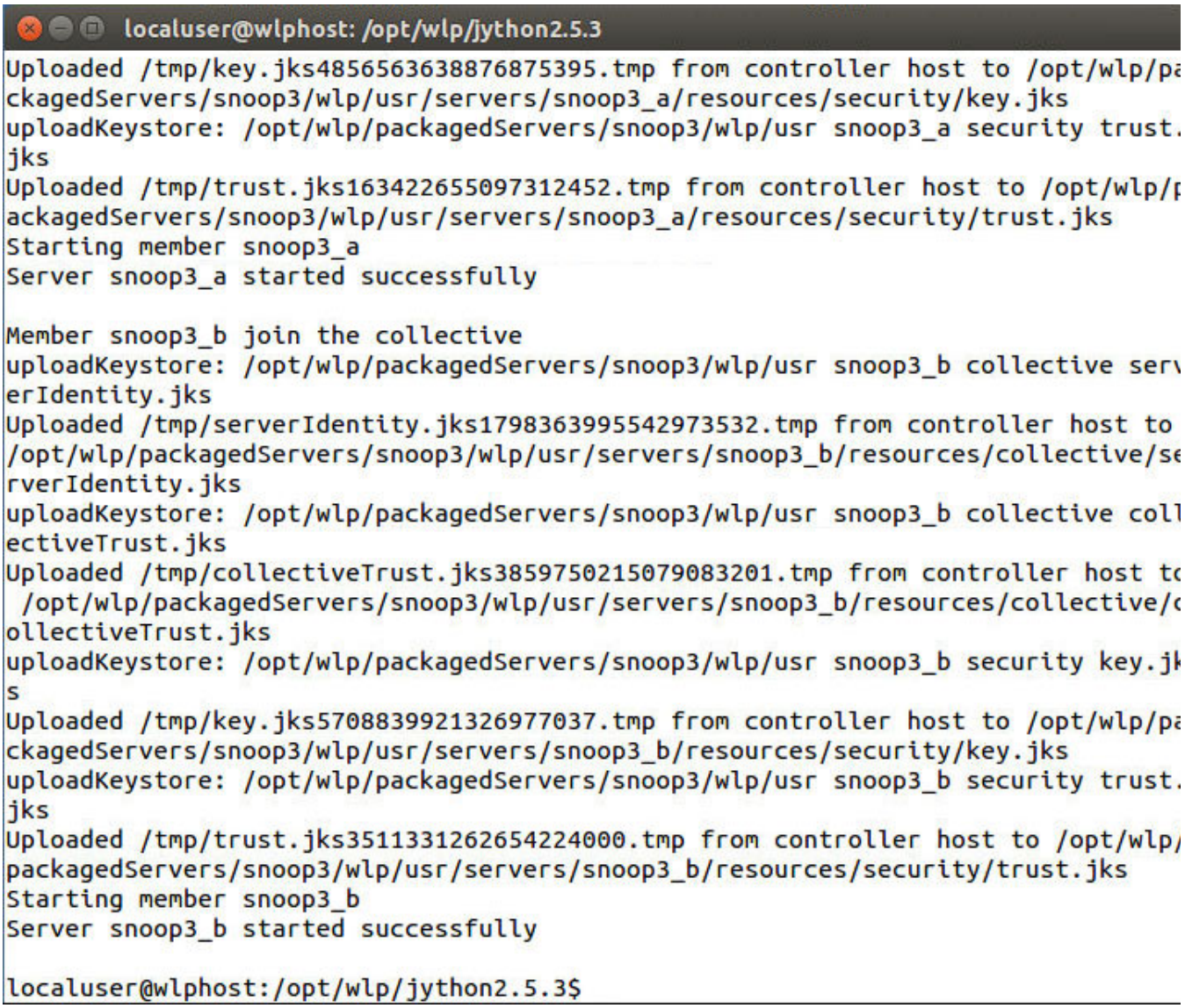
__ a. Go to the `/opt/wlp/jython2.5.3` directory.

__ b. Deploy the snoop3 application by using the `deployMembers.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/deployMembers.py
--zipFile=/opt/labfiles/management/snoop3/snoop3.zip
--installDir=/opt/wlp/packagedServers/snoop3 --runtimeDir=/opt/wlp
--installHost=wlphost --rpcUser=localuser --rpcUserPassword=passw0rd
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --host=wlphost --port=9443 --user=admin
--password=passw0rd
```

Note

The screen capture shows the last part of successfully running the `deployMembers.py` script.



__ 3. Verify the deployed application.

__ a. Maximize the web browser.

__ a. Go to member1, snoop3_a. Use the **httpPort** number that you noted earlier from the `bootstrap.properties` file. Enter the following URL:

`http://wlphost:9186/snoop2/SnoopServlet`

__ b. Go to member2, snoop3_b. Use the **httpPort** number that you noted earlier from the `bootstrap.properties` file. Enter the following URL:

`http://wlphost:9187/snoop2/SnoopServlet`

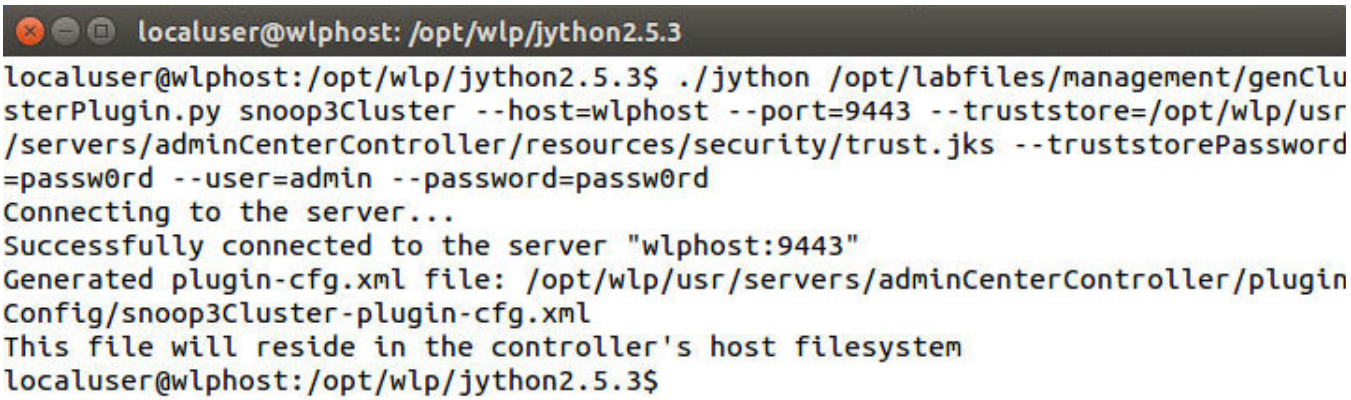
__ c. Minimize the browser window.

__ 4. Generate the plug-in configuration.

__ a. Go to the terminal window.

__ b. Generate an updated plug-in configuration by using the `genClusterPlugin.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/genClusterPlugin.py snoop3Cluster
--host=wlphost --port=9443
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/trust
.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```



__ 5. Merge the plug-in configuration files.

__ a. The collective controller does not yet support generation of a plug-in configuration for multiple clusters. You need to merge them manually To save time, a merged copy of the file is provided at `/opt/labfiles/management/snoopCombined-plugin-cfg.xml`. Go to the `/opt/labfiles/management` directory.

Information

The merged copy of the plug-in file contains the following changes that are compared to the one you generated:

- A new server cluster with four members is used to route across the two clusters.
- A new `LoadBalanceWeight` attribute is added for each server, and set to 10 for the two existing servers, and to 0 for the new servers. This means that the web server will still load balance equally to the existing servers, and will not route any request to the new servers yet.

__ b. Copy the merged plug-in file and replace the existing `snoop2Cluster-plugin-cfg.xml` file in the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory. To copy the updated file, enter the following command:

```
cp -f snoopCombined-plugin-cfg.xml
/opt/IBM/WebSphere/Plugins/config/IHS1/snoop2Cluster-plugin-cfg.xml
```

__ 6. Restart the IBM HTTP Server

Information

When you copy the merged file and overwrite the `snoop2Cluster-plugin-cfg.xml` file, the existing configuration is replaced and the web server should pick up the new changes almost right away. The default setting is 60 seconds. For this exercise, you restart the web server instead of waiting.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ c. Enter the following command to start the HTTP Server:

```
./apachectl start
```

__ 7. Verify the configuration.

__ a. Maximize the browser window.

__ b. Go to the following URL:

`http://wlphost:9180/snoop2/SnoopServlet`

Request Information:

Request method	GET
Request URI	/snoop2/SnoopServlet
Request protocol	HTTP/1.1
Servlet path	/SnoopServlet
Path info	<none>
Path translated	<none>
Character encoding	<none>
Query string	<none>
Content length	<none>
Content type	<none>
Server name	wlphost
Server port	9180
Remote user	<none>
Remote address	127.0.0.1
Remote host	127.0.0.1
Remote port	52188
Local address	127.0.1.1
Local host	wlphost
Local port	9185
Authorization scheme	<none>

__ c. Refresh the browser a few times to show which servers are processing requests. These servers that can process the requests are:

- snoop2_a: Port 9184
- snoop2_b: Port 9185

__ d. Minimize the browser window.

__ 8. Edit the `snoop2Cluster-plugin-cfg.xml` file and modify the LoadBalanceWeight settings.

__ a. In the terminal window, go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ b. Open the `snoop2Cluster-plugin-cfg.xml` file by using an editor such as vi or gedit.

__ c. Make the following changes:

- For the server with port 9184, set the value of **LoadBalanceWeight=0**

- For the server with port 9187, set the value of **LoadBalanceWeight=10**

*snoop2Cluster-plugin-cfg.xml

```
            </Transport>
        </Server>
           <Server LoadBalanceWeight="0" CloneID="4d01018d-0c6c-4c77-85b4-
f57ba712ccb7" ConnectTimeout="5" ExtendedHandshake="false"
MaxConnections="-1" Name="default_node_defaultServer_0_0"
ServerIOTimeout="900" WaitForContinue="false">
            <Transport Hostname="wlphost" Port="9184" Protocol="http"/>
            <Transport Hostname="wlphost" Port="9547" Protocol="https">
               <Property Name="keyring" Value="keyring.kdb"/>
               <Property Name="stashfile" Value="keyring.sth"/>
               <Property Name="certLabel" Value="LibertyCert"/>
            </Transport>
        </Server>
              <Server LoadBalanceWeight="10"
CloneID="fa2ae16f-56bf-4d80-846e-2ee7a2f9e4ad" ConnectTimeout="5"
ExtendedHandshake="false" MaxConnections="-1"
Name="default_node_defaultServer_1_1" ServerIOTimeout="900"
WaitForContinue="false">
            <Transport Hostname="wlphost" Port="9187" Protocol="http"/>
            <Transport Hostname="wlphost" Port="9550" Protocol="https">
               <Property Name="keyring" Value="keyring.kdb"/>
               <Property Name="stashfile" Value="keyring.sth"/>
               <Property Name="certLabel" Value="LibertyCert"/>
            </Transport>
```

Information

When you set `LoadBalanceWeight=0`, only requests within the same session are routed to that application server whose weight you set to zero. Typically you want to wait for all sessions to be completed, or timed out, before you decommission the server. Which in this case, is snoop2_a. Since the snoop application does not use sessions, the plug-in is able to redirect the requests immediately between snoop2_b and snoop3_b.

__ d. Save the changes.

__ e. Close the file when completed.

__ 9. Test the changes to the configuration.

__ a. Maximize the Firefox web browser.

__ b. Reload the browser a few times, and verify that snoop3_b (port 9187) and snoop2_b (port 9185) are the servers that are receiving requests.

__ c. Minimize the browser window.

Note

If you get an error that you are unable to the load page, stop and restart the IBM HTTP Server.

__ 10. Edit the `snoop2Cluster-plugin-cfg.xml` file and modify the LoadBalanceWeight settings.

__ a. In the terminal window, go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ b. Open the `snoop2Cluster-plugin-cfg.xml` file by using an editor such as vi or gedit.

__ c. Make the following changes:

- For the server with port 9185, set the value of **LoadBalanceWeight=0**
- For the server with port 9186, set the value of **LoadBalanceWeight=10**

__ d. Save the changes.

__ e. Close the file when completed.

__ 11. Test the changes to the configuration.

__ a. Maximize the Firefox web browser.

__ b. Reload the browser a few times and verify that only snoop3_a and snoop3_b are processing the requests.

Optional

Rollback by gradually setting the weight to 10 for the original application servers, and to 0 for the new application servers.

__ c. When completed, close the browser window.

Part 7: Cleaning up the environment

In this part of the exercise, you clean up the deployed artifacts.

__ 1. Stop the IBM HTTP Server.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ 2. Undeploy the servers.

__ a. Go to the `/opt/wlp/jython2.5.3` directory.

__ b. Stop the snoop2 cluster with the `stopCluster.py` script. To use the script, enter the following command:

```
./jython /opt/labfiles/management/stopCluster.py --clusterName=snoop2Cluster
--host=wlphost --port=9443
```

```
--truststore=/opt/wlp/usr/servers/adminCenterController/resources/security/tr
ust.jks --truststorePassword=passw0rd --user=admin --password=passw0rd
```

__ c. Go to the `/opt/wlp/bin` directory.

__ d. Set the `WLP_USER_DIR` environment variable as the servers are in a non-default directory. To set the variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/packagedServers/snoop2/wlp/usr
```

__ e. Remove the snoop2 members from the collective controller. To remove snoop2_a, enter the following command:

```
./collective remove snoop2_a --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ f. To remove snoop2_b, enter the following command:

```
./collective remove snoop2_b --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ g. Set the `WLP_USER_DIR` environment variable for the snoop3 server. To set the environment variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/packagedServers/snoop3/wlp/usr
```

__ h. Remove the snoop3 members from the collective controller. To remove snoop3_a, enter the following command:

```
./collective remove snoop3_a --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ i. Remove the snoop3 members from the collective controller. To remove snoop3_b, enter the following command:

```
./collective remove snoop3_b --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ 3. Remove the directories and update an environment variable.

__ a. Go to the `/opt/wlp/packagedServers/` directory.

__ b. Remove the `snoop2` and `snoop3` directories.

__ c. Change the `WLP_USER_DIR` environment variable back to the default directory. To change the environment variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/usr
```

__ 4. Stop the server adminCenterController.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop adminCenterController
```

__ c. Exit the terminal window.

End of exercise

Exercise review and wrap-up

In this exercise, you used command line and Jython scripting tools to deploy and maintain clustered application servers centrally through the collective controller.

Exercise 3. Dynamic Routing

Estimated time

01:15

Overview

In this exercise, you use the Dynamic Routing feature of Liberty to enable routing of HTTP requests to collective members without having to regenerate the WebSphere plug-in configuration file when the environment changes.

Objectives

After completing this exercise, you should be able to:

- Dynamically route requests from the WebSphere plug-in to static clusters
- Configure separate HTTP ports for administration and applications

Introduction

When servers, cluster members, applications, or virtual hosts are added, removed, started, stopped, or modified, the new information is dynamically delivered to the WebSphere plug-in. Requests are routed based on up-to-date information. The feature provides the dynamic routing service, which dynamically retrieves routing information from the collective repository and delivers this information to the WebSphere plug-in.

Requirements

To complete this exercise, you need the WebSphere Liberty binary files and Liberty installed. You also need the collective controller, adminCenterController, created in the Managing Liberty collectives with the Admin Center exercise.

Exercise instructions

Routing of web requests to servers in a Liberty collective is done by using a web server with the WebSphere plug-in. With static routing, the information that is used to route requests is read from a plug-in configuration file. The routing information in the file contains the endpoint information of the servers in the collective. The routing information for each server is generated by invoking an administrative MBean method on each server.

For multiple servers, the routing information for the WebSphere plug-in must be merged. The WebSphere plug-in detects when a server or application is unavailable, or when communication errors occur with the server or application. This topology becomes more complicated as servers or applications are added. The routing information in the file must be regenerated for each change. The changes must then be merged into the configuration file that is provided to the WebSphere plug-in.

The dynamic routing feature enables routing of HTTP requests to members of Liberty collectives without having to regenerate the WebSphere plug-in configuration file when the environment changes. When servers, cluster members, applications, or virtual hosts are added, removed, started, stopped, or modified, the new information is dynamically delivered to the WebSphere plug-in. Requests are routed based on up-to-date information.

To use the dynamic routing feature, you need to configure the `dynanamicRouting-1.0` feature in the `server.xml` file on the controller. In this part of the exercise, you configure dynamic routing for a Liberty collective.

Part 1: Configuring dynamic routing

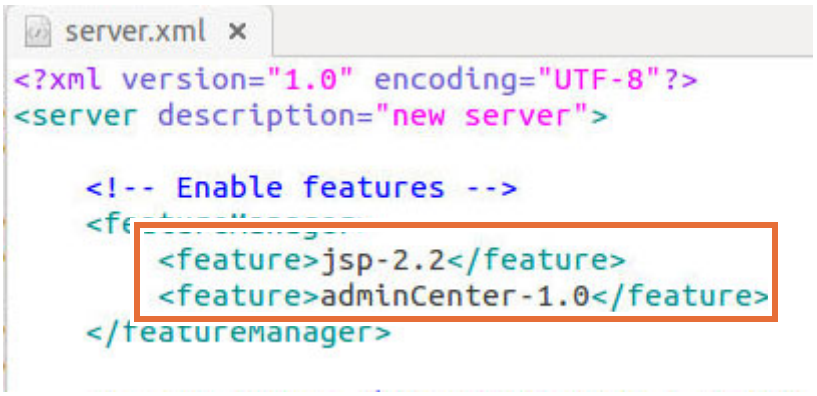
__ 1. Configure dynamic routing in the collective controller.

__ a. Open a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

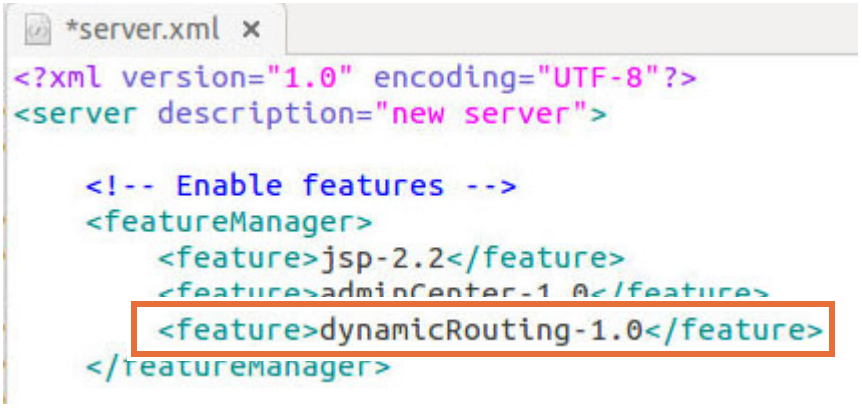
__ c. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Go to the feature manager section. You can see the two features that are enabled are `jsp-2.2` and `adminCenter-1.0`.



__ e. Add the following feature:

```
<feature>dynamicRouting-1.0</feature>
```



The `dynamicRouting-1.0` feature is required in collective controllers only.

Information

The dynamic routing feature is an Intelligent Management feature of the WebSphere plug-in for Apache and IBM HTTP Server that provides On Demand Router capabilities for the plug-in. The dynamic routing feature enables a server to run a REST service to which the plug-in can connect to dynamically route to all servers in a collective.

__ f. Save the changes.

__ g. Close the file.

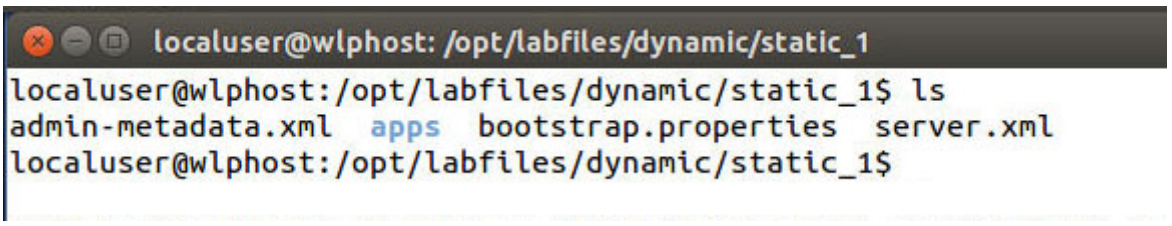
__ 2. Examine the customized files for the server static_1.

__ a. Go to the `/opt/labfiles/dynamic` directory.

__ b. List the contents of the directory. You can see a directory for static_1 and static_2. Each directory contains the customized files for a Liberty server runtime environment.

__ c. Go to the `static_1` directory.

__ d. List the contents of the directory. You can see the customized files for the server and an apps directory that contains an application.



__ e. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ f. Go to the featureManager section. Notice that the file includes the following features:

- **collectiveMember-1.0**: Which indicates that the controller manages the server
- **clusterMember-1.0**: Which indicates that the server is to be a member of a cluster

server.xml

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>collectiveMember-1.0</feature>
        <feature>clusterMember-1.0</feature>
    </featureManager>

    <!-- defaultHttpEndpoint is used for REST connector only -->
    <httpEndpoint id="defaultHttpEndpoint"
        host="wlphost"
        httpPort="${httpAdminPort}"
        httpsPort="${httpsAdminPort}" />

    <!-- use ApplicationHttpEndpoint for application serving -->
    <httpEndpoint id="ApplicationHttpEndpoint"
        host="wlphost"
        httpPort="${httpPort}"
        httpsPort="${httpsPort}" />
```

__ g. Go to the next section. You can see separate httpEndpoint details for administration and application traffic.

server.xml

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>collectiveMember-1.0</feature>
        <feature>clusterMember-1.0</feature>
    </featureManager>

    <!-- defaultHttpEndpoint is used for REST connector only -->
    <httpEndpoint id="defaultHttpEndpoint"
        host="wlphost"
        httpPort="${httpAdminPort}"
        httpsPort="${httpsAdminPort}" />

    <!-- use ApplicationHttpEndpoint for application serving -->
    <httpEndpoint id="ApplicationHttpEndpoint"
        host="wlphost"
        httpPort="${httpPort}"
        httpsPort="${httpsPort}" />
```

__ h. Go to the virtual host section. You can see separate virtualHost details for administration and application traffic.

server.xml

```
    <httpEndpoint id="ApplicationHttpEndpoint"
        host="wlphost"
        httpPort="${httpPort}"
        httpsPort="${httpsPort}" />

    <!-- Use virtual host default_host for defaultHttpEndpoint -->
    <virtualHost id="default_host"
allowFromEndpointRef="defaultHttpEndpoint" />

    <!-- use virtual host application_host for applications -->
    <virtualHost id="application_host"
allowFromEndpointRef="ApplicationHttpEndpoint" >
        <hostAlias>wlphost:${httpPort}</hostAlias>
        <hostAlias>wlphost:${httpsPort}</hostAlias>
        <hostAlias>${webServerHost}:${webServerPort}</hostAlias>
    </virtualHost>

   <!-- Generate plugin only for ApplicationHttpEndpoint -->
    <pluginConfiguration httpEndpointRef="ApplicationHttpEndpoint" />

   <webApplication id="static" location="static.war"  />
```

__ i. When completed, close the file.

__ j. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.

bootstrap.properties

```
httpPort=9188
httpsPort=9551
httpAdminPort=9088
httpsAdminPort=9451
webServerHost=*
webServerPort=9180
```

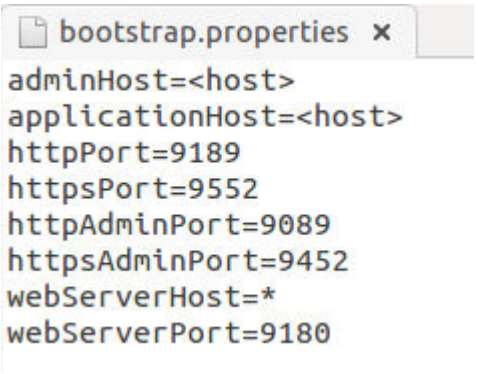
You can see the ports that are defined for this server. Note the httpPort is 9188 and the webServerPort is 9180.

__ k. When completed, close the file.

__ 3. Examine the customized files for the server static_2.

__ a. Go to the `/opt/labfiles/dynamic/static_2` directory and list the contents of the directory. You can see the same files, each customized for static_2.

__ b. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.



You can see the ports that are defined for this server. Note the httpPort is 9189 and the webServerPort is 9180.

__ c. When completed, close the file.

__ 4. Copy the `static_1` directory to the `servers` directory.

__ a. Go to the `/opt/labfiles/dynamic` directory.

__ b. Copy the `static_1` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

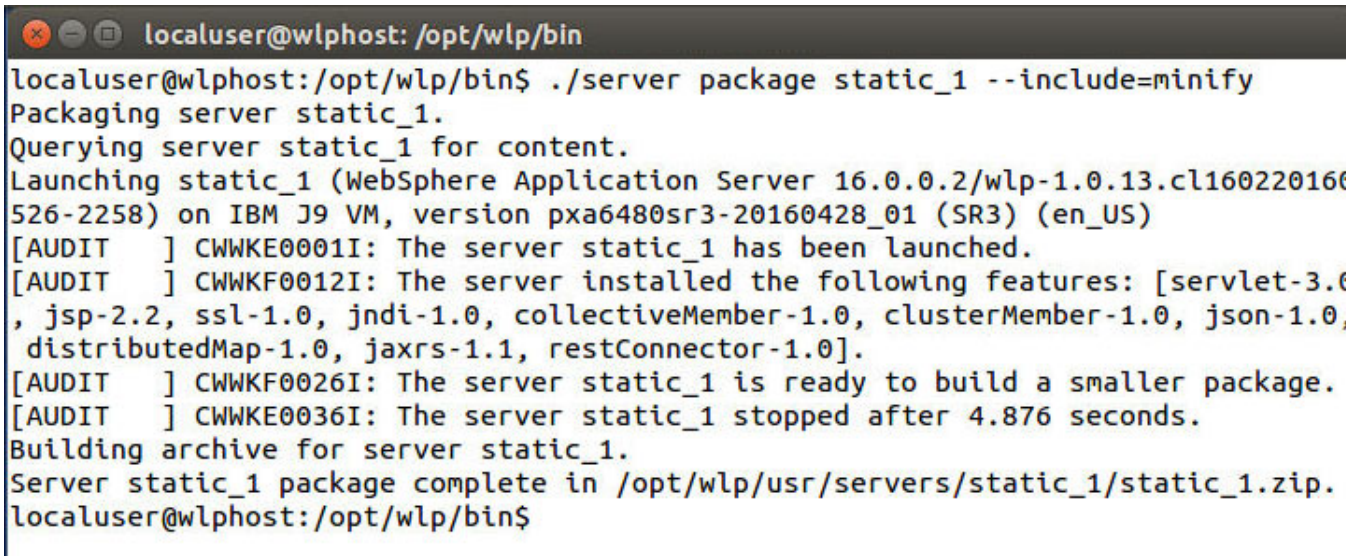
```
cp -r static_1 /opt/wlp/usr/servers/
```

__ 5. Create a packaged server. From the command line, you can create a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To create the packaged server `static_1`, enter the following command:

```
./server package static_1 --include=minify
```



Wait until you see that the package is complete. You can also see that this command creates a file, `static_1.zip`, in the `static_1` directory.

__ 6. Start and update the collective controller.

__ a. To start the collective controller, enter the following command:

```
./server start adminCenterController
```

__ b. If needed, you can verify the status of the server. To verify that the collective controller is started, enter the following command:

```
./server status adminCenterController
```

__ c. Update the controller with more parameters that are required when you deploy a packaged server. In addition to the Liberty admin user and password, you provide details on the host computer user and password. The option `rpcUser` is the user ID you use to log in to the operating system, and `rpcUserPassword` is the corresponding password.

To update the controller, enter the following command:

```
./collective updateHost wlphost --host=wlphost --port=9443 --user=admin
--password=passw0rd --rpcUser=localuser --rpcUserPassword=passw0rd
--hostReadPath=/opt/wlp/packagedServers
--hostWritePath=/opt/wlp/packagedServers
```

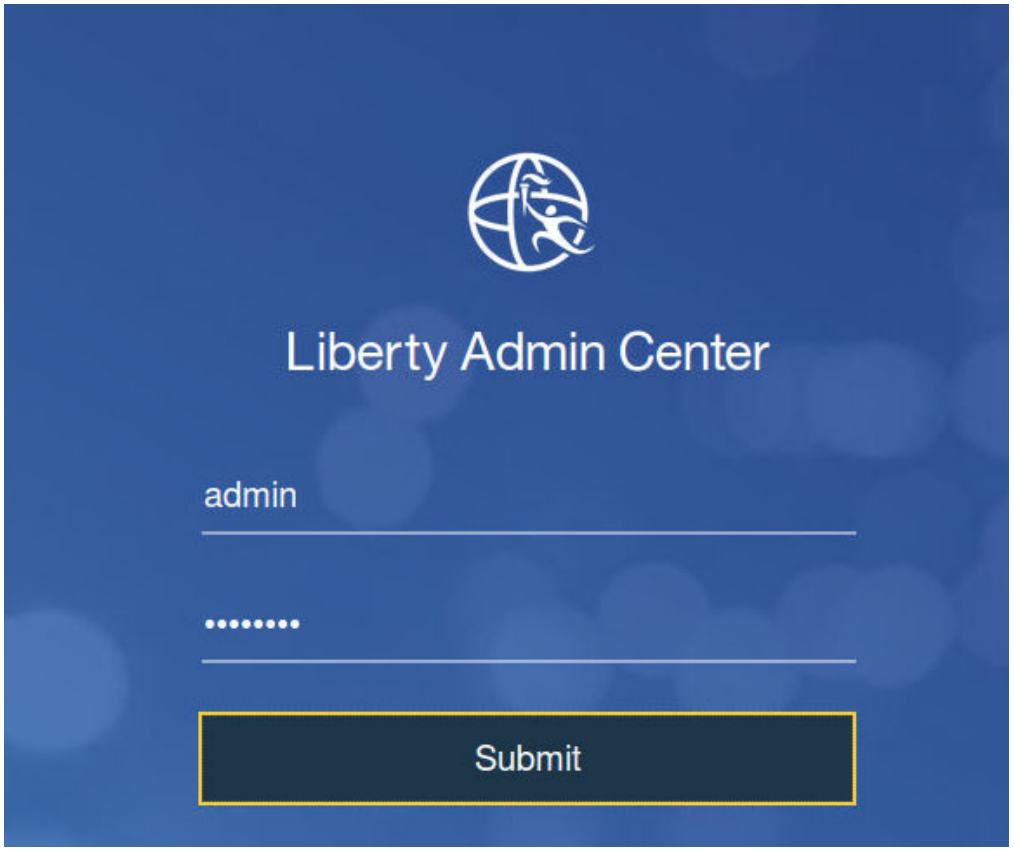
When prompted, accept the certificate chain.

__ 7. Start the Admin Center console.

__ a. Open the Firefox web browser and go to the following website:

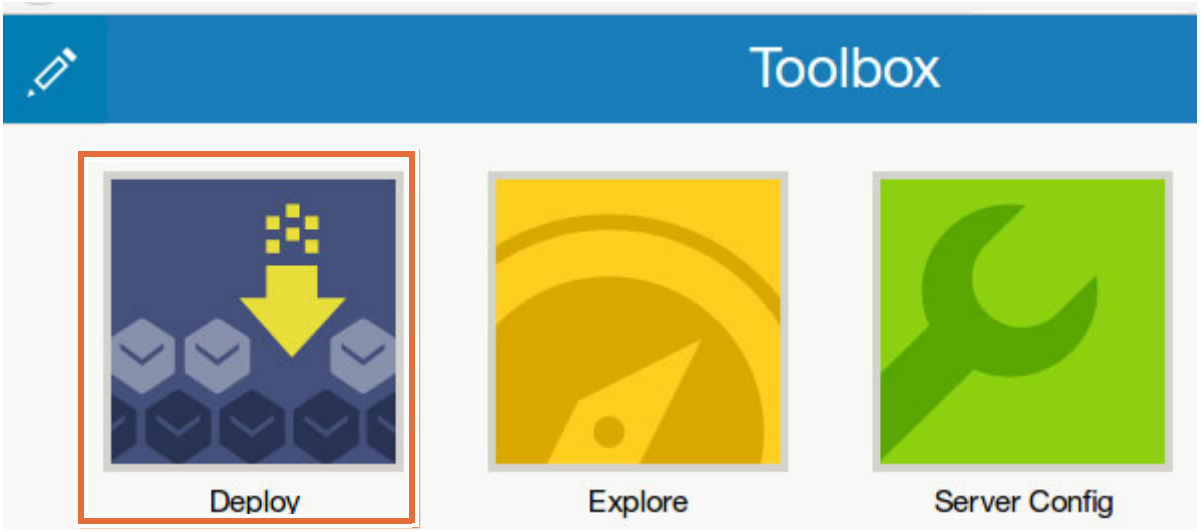
```
http://wlphost:9080/adminCenter
```

__ b. In the login area, enter `admin` as the User Name and `passw0rd` as the Password and click **Submit.**



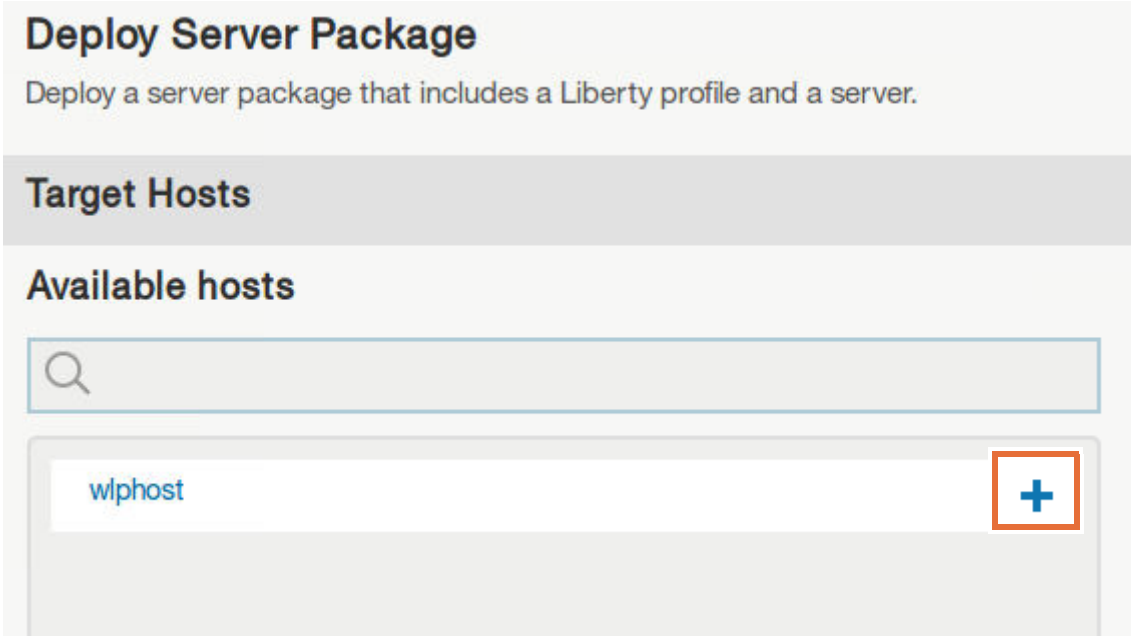
__ 8. Deploy a packaged server by using the Admin Center.

__ a. In the Admin Center, click **Deploy**.



__ b. Go to the Available hosts section.

__ c. Select the host that you want to deploy the package to. Next to wlphost, click **+**.



You can see that the host, wlphost, is moved to the Selected hosts section.

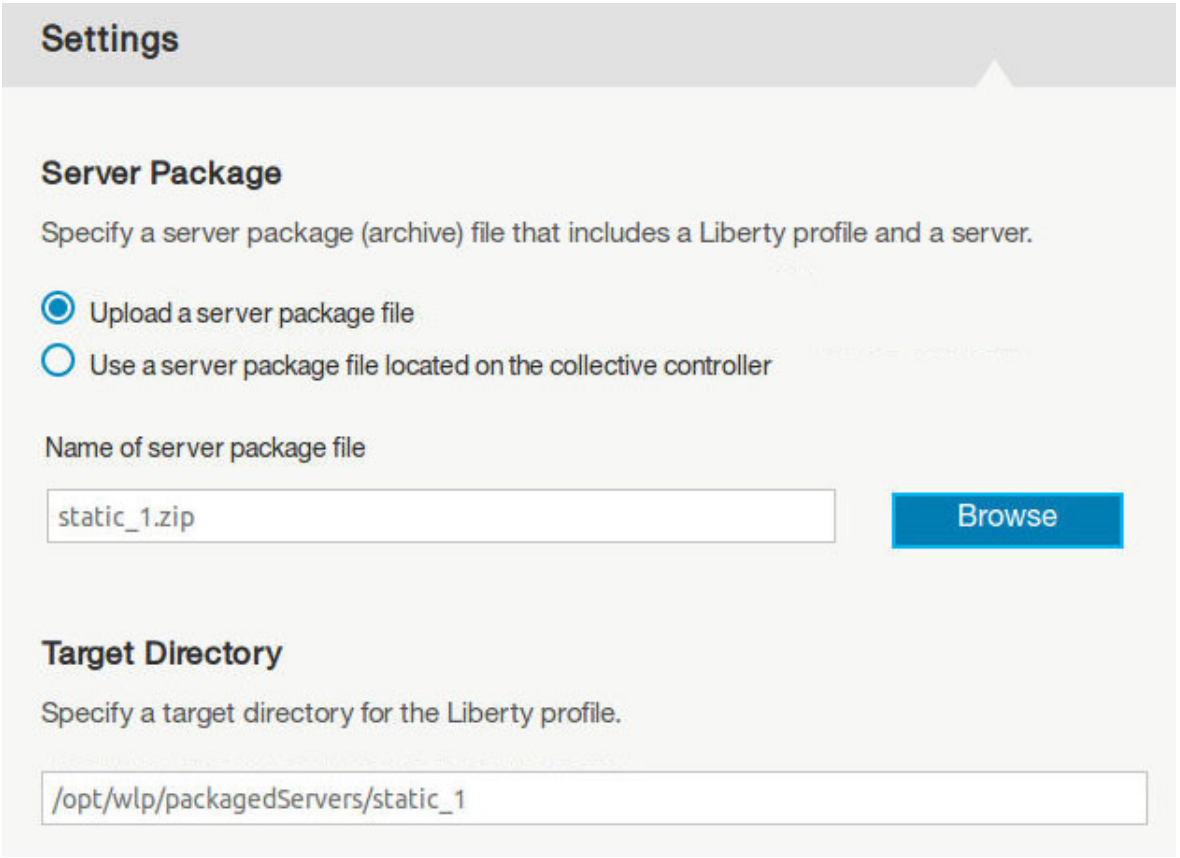
__ d. Go to the Settings section.

__ e. Under Server Package, verify that the option **Upload a server package file** is selected.

__ f. Click **Browse,** select `/opt/wlp/usr/servers/static_1/static_1.zip`, and click **Open**.

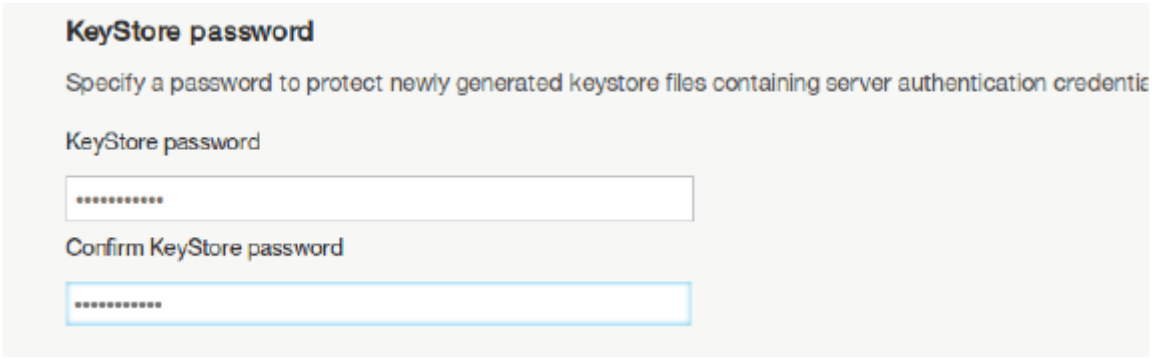
__ g. In the **Target Directory** field, enter the following directory:

`/opt/wlp/packagedServers/static_1`



__ h. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`

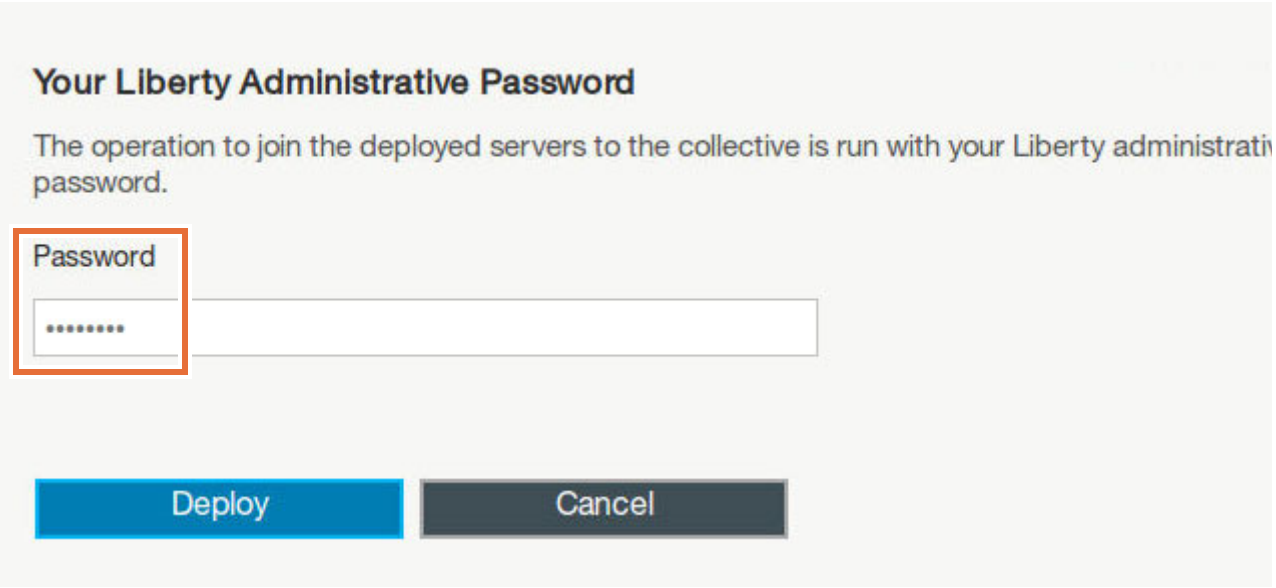


__ i. Go to the Remote Management Credentials section.

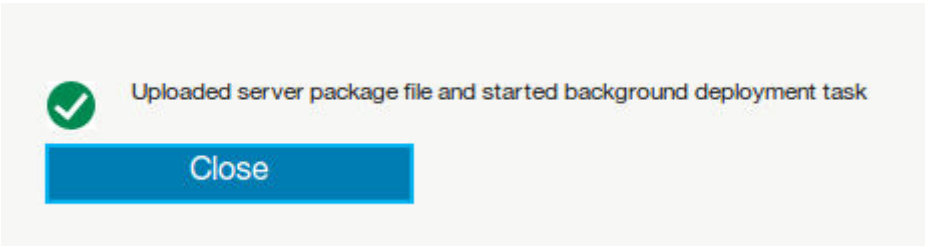
__ j. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

__ k. Go to the Your Liberty Administrative Password section.

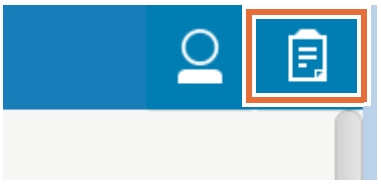
__ l. In the **Password** field, enter: `passw0rd`



__ m. At the bottom of the page, click **Deploy.** Wait for the message that the file is uploaded and a background deployment task is started.



__ n. To see the Task History, click the **Background Tasks** icon at the upper-right corner of the page.



__ o. Click **Task Details and History** to see the status of the deployment.

__ p. The green check mark and green progress bar indicate a successful deployment. Click **Deploy Installation - static_1 > wlphost** to view the details of the task.

__ q. Minimize the Admin Center browser.

__ 9. Examine what the collective controller deployed.

__ a. Go to the terminal window.

__ b. Go to the `/opt/wlp/packagedServers/static_1/wlp/usr/servers/static_1` directory.

__ c. List the contents of the directory. Notice that the collective controller creates the `collective-join-include.xml` file that allows the member to join with the controller.

__ d. Open the `collective-join-include.xml` file with an editor such as vi or gedit. Examine the settings in the file. You can see the details about the collective controller.

collective-join-include.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<server description="This file was generated by the 'collective join'
command on 2016-09-28 06:36:31 EDT.">
    <featureManager>
        <feature>collectiveMember-1.0</feature>
    </featureManager>

    <!-- Define the host name for use by the collective.
         If the host name needs to be changed, the server should be
         removed from the collective and re-joined or re-replicated. --
>
    <variable name="defaultHostName" value="wlphost" />

    <!-- Remote host authentication configuration -->
    <hostAuthInfo useHostCredentials="true" />

    <!-- Connection to the collective controller -->
    <collectiveMember controllerHost="wlphost"
                      controllerPort="9443" />
```

__ e. When completed, close the file.

__ 10. Repeat the prior steps to package and deploy the static_2 packaged server.

__ a. Go to the `/opt/labfiles/dynamic` directory.

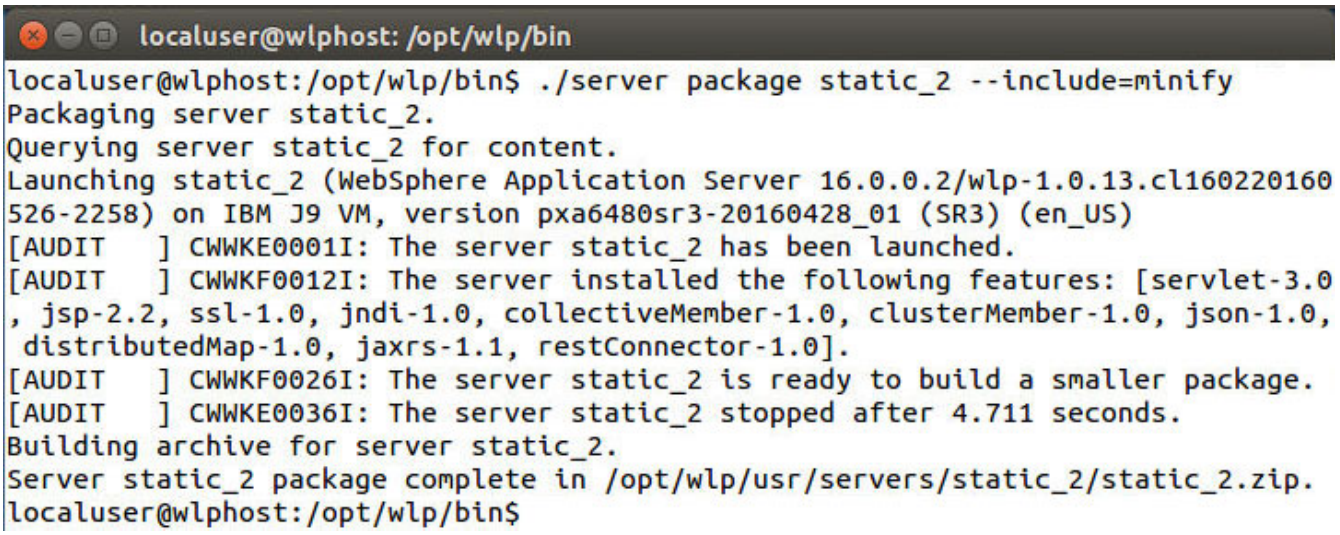
__ b. Copy the `static_2` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

```
cp -r static_2 /opt/wlp/usr/servers/
```

__ c. Go to the `/opt/wlp/bin` directory.

__ d. To create the packaged server `static_2`, enter the following command:

```
./server package static_2 --include=minify
```



__ e. Maximize the Admin Center.

__ f. In the Admin Center, click the **Toolbox** icon to go back to the toolbox.

__ g. Click **Deploy**.

__ h. Go to the Available hosts section. Select the host that you want to deploy the package to. Next to wlphost, click **+**.

__ i. Go to the Settings section. Under Server Package, verify that the option **Upload a server package file** is selected.

__ j. Click **Browse,** select `/opt/wlp/usr/servers/static_2/static_2.zip`, and click **Open**.

__ k. In the **Target Directory** field, enter the following directory:

```
/opt/wlp/packagedServers/static_2
```

__ l. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`

__ m. Go to the Remote Management Credentials section. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

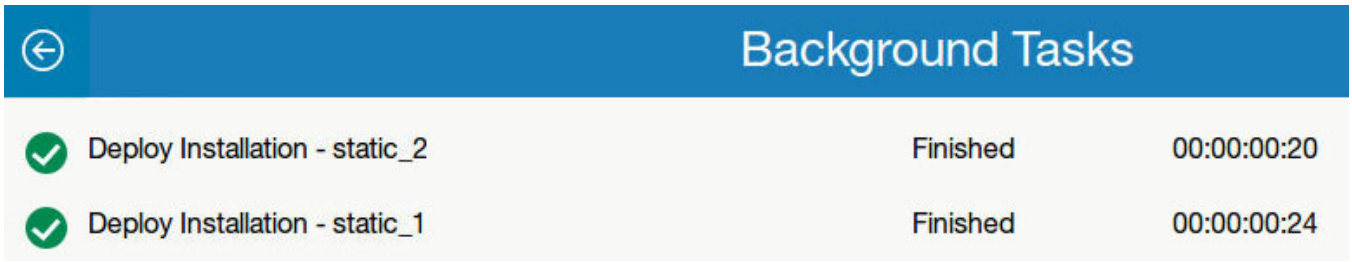
__ n. Go to the Your Liberty Administrative Password section. In the **Password** field, enter: `passw0rd`

__ o. At the bottom of the page, click **Deploy.** Wait for the message that the file is uploaded and a background deployment task is started.

__ p. To see the Task History, click the **Background Tasks** icon at the upper-right corner of the page.

__ q. Click **Task Details and History** to see the status of the deployment.

__ r. The green check mark and green progress bar indicate a successful deployment.



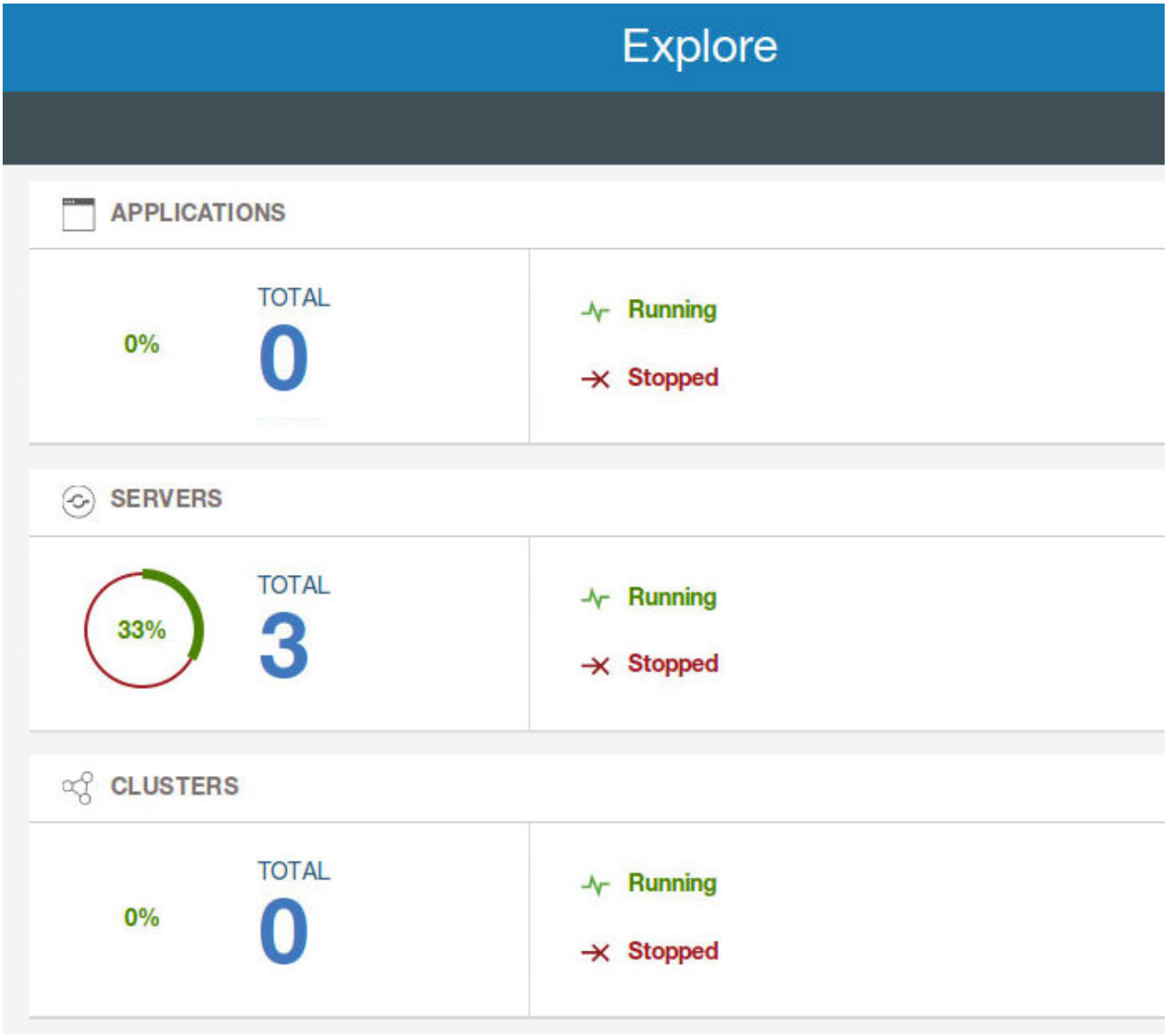
Part 2: Exploring the configuration

__ 1. Explore the configuration.

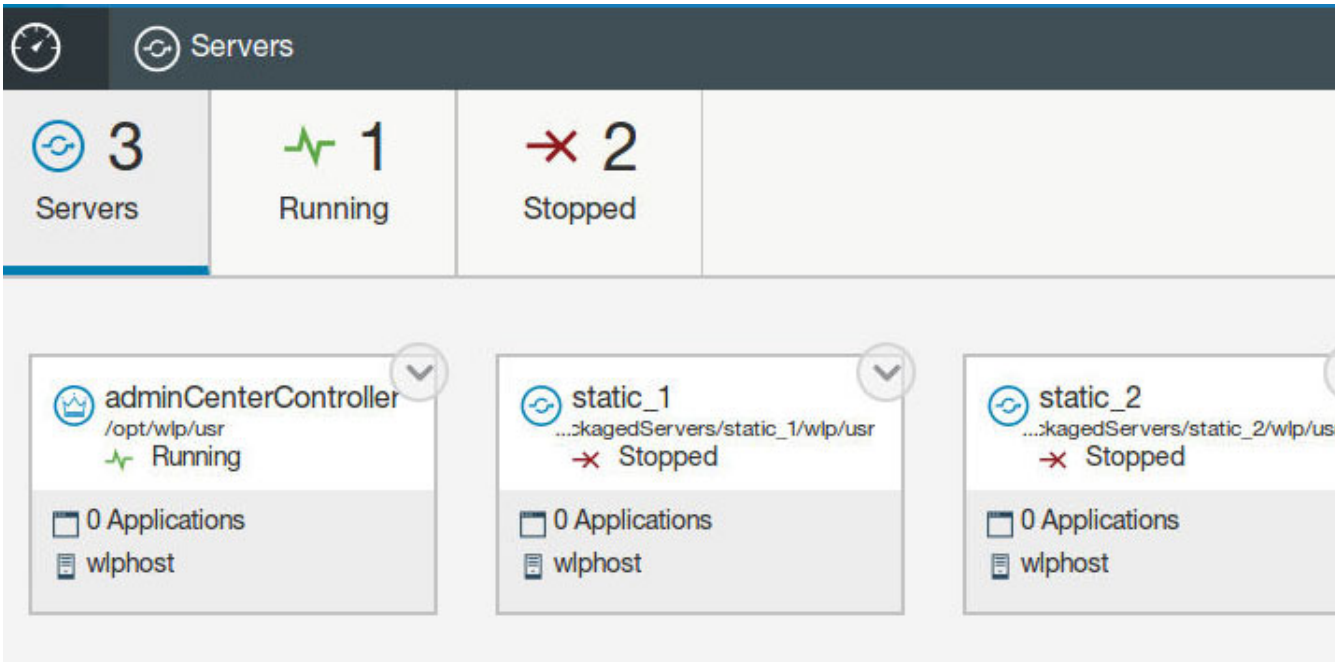
__ a. Click the **Toolbox** icon to go back to the toolbox.

__ b. Click **Explore**.

__ c. Go to the SERVERS section. Notice that there are now two new servers. However, no cluster is detected yet.



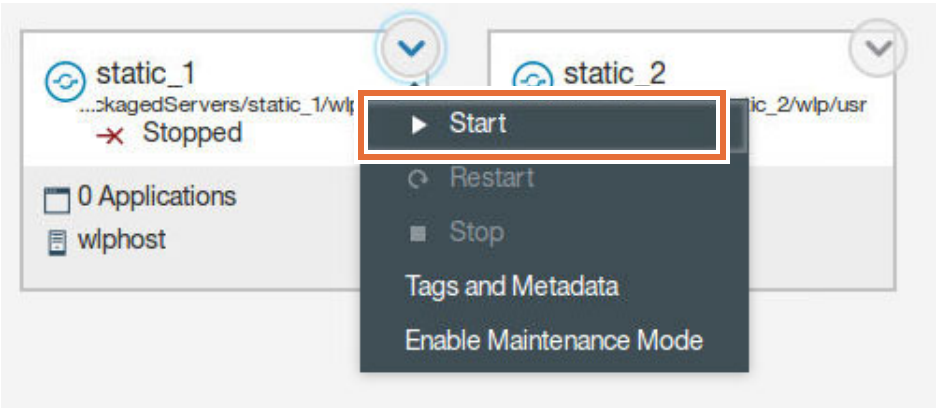
__ d. Click the **SERVERS** pane. You can see static_1 and static_2 are in the stopped state.



__ 2. Start the servers.

__ a. Click the **Actions** icon (down-arrow) in the upper right of the static_1 server box.

__ b. Then, click **Start** to start the static_1 server.



__ c. In the start server static_1 window, click **Start**.

__ d. After a few minutes, the status of the server changes to **Running**.

__ a. Click the **Actions** icon (down-arrow) in the upper right of the static_2 server box.

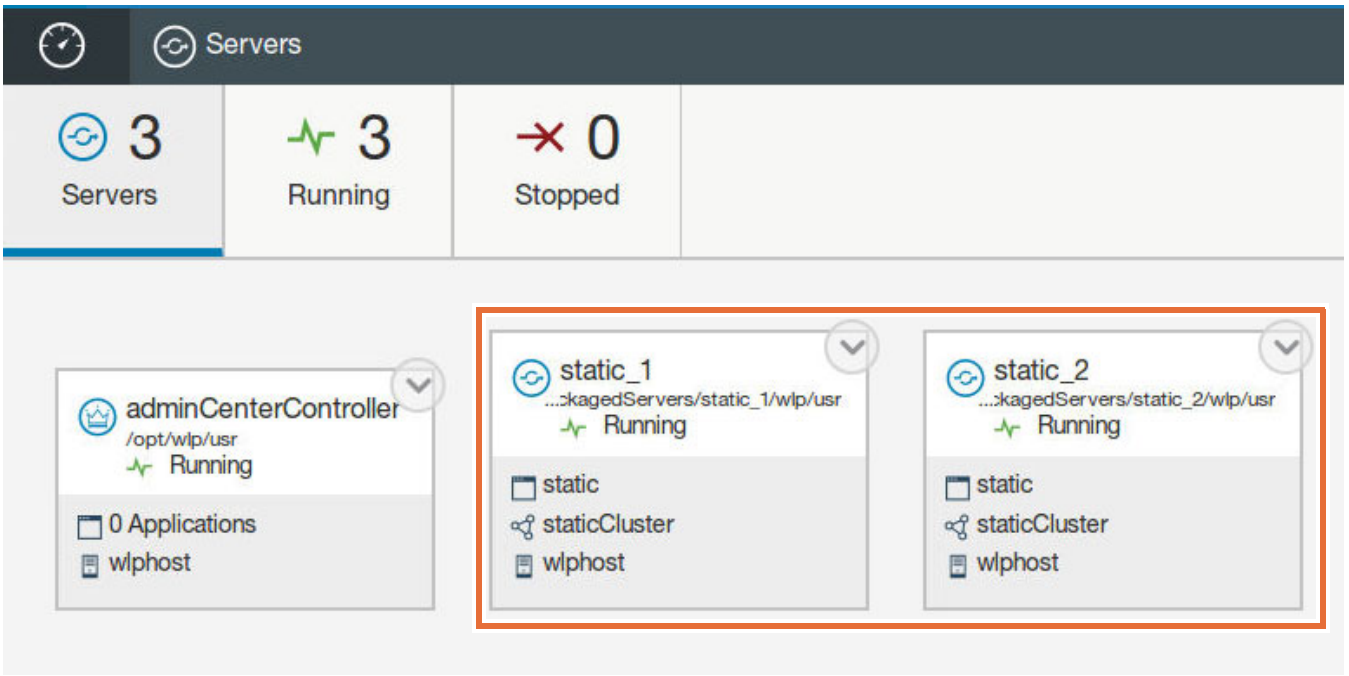
__ b. Then, click **Start** to start the static_2 server.

__ c. In the start server static_2 window, click **Start**.

__ d. After a few minutes, the status of the server changes to **Running**.

__ 3. Verify the configuration.

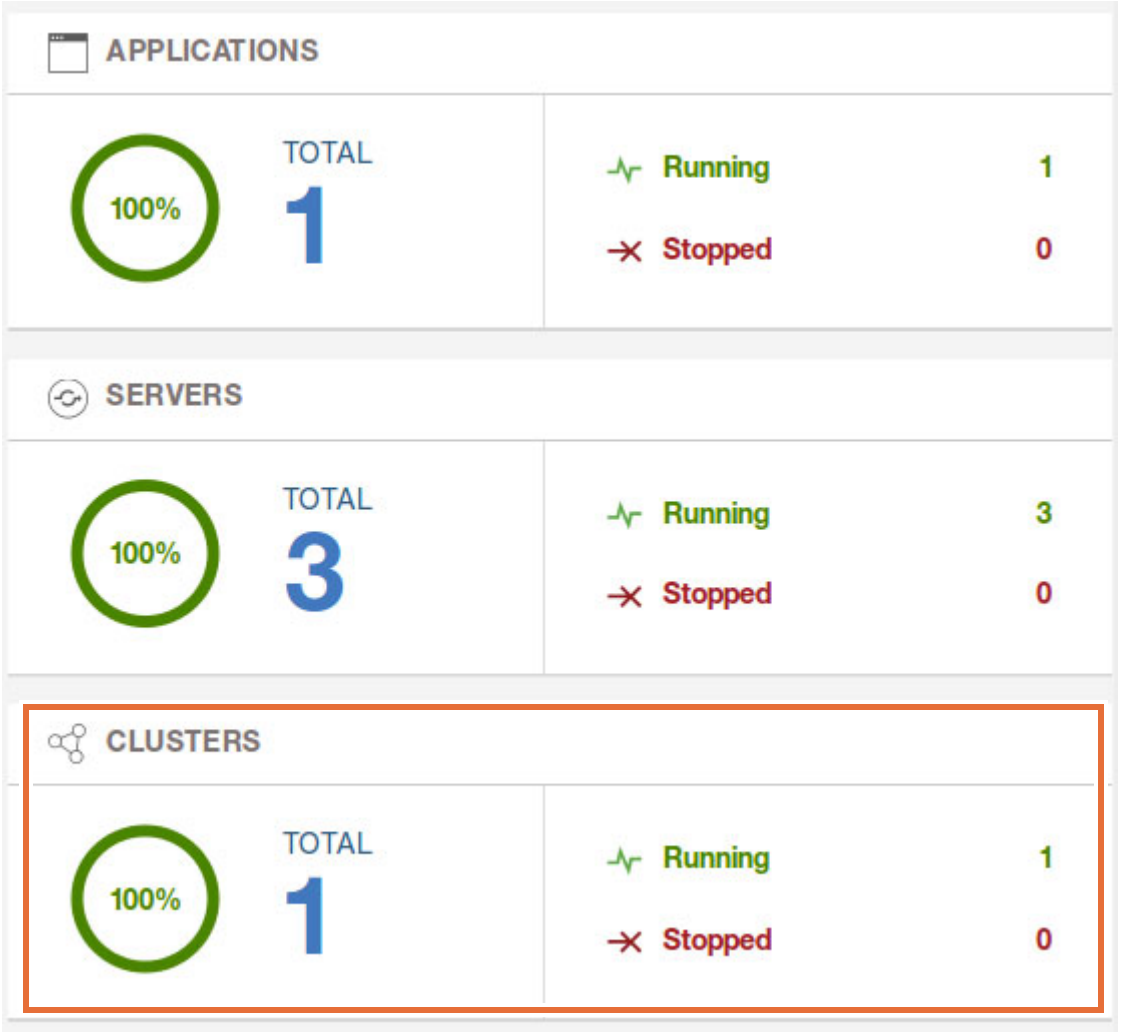
__ a. Verify that both servers are running.



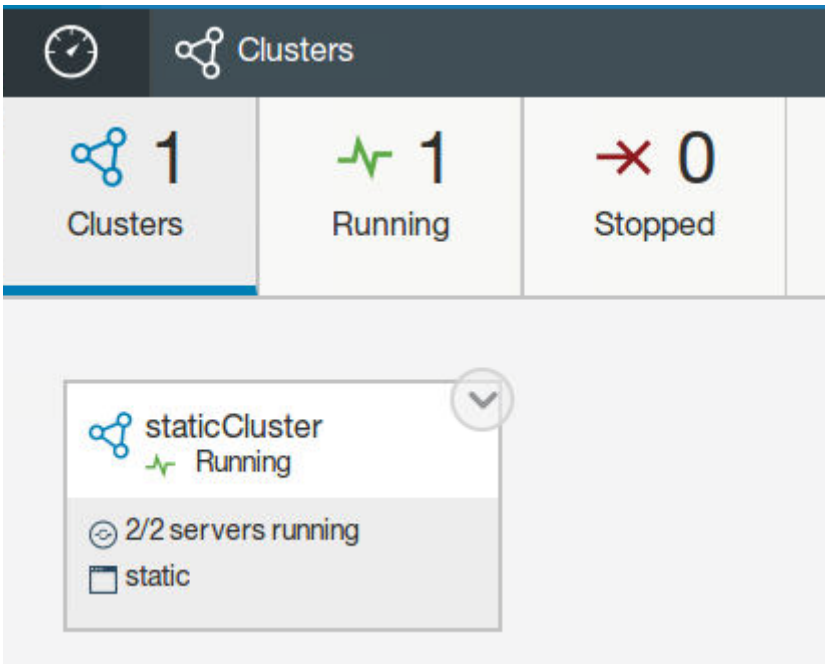
__ b. Click the **Dashboard** icon to go back to the dashboard.

__ c. Go to the SERVERS section. You can see that three servers are running.

__ d. Go to the CLUSTERS section. A cluster is now detected and an application is running.



__ e. Click the **CLUSTERS** pane. You can see the cluster, staticCluster is running, with two servers in the cluster. Both servers are running. The cluster also includes one application, the static application.



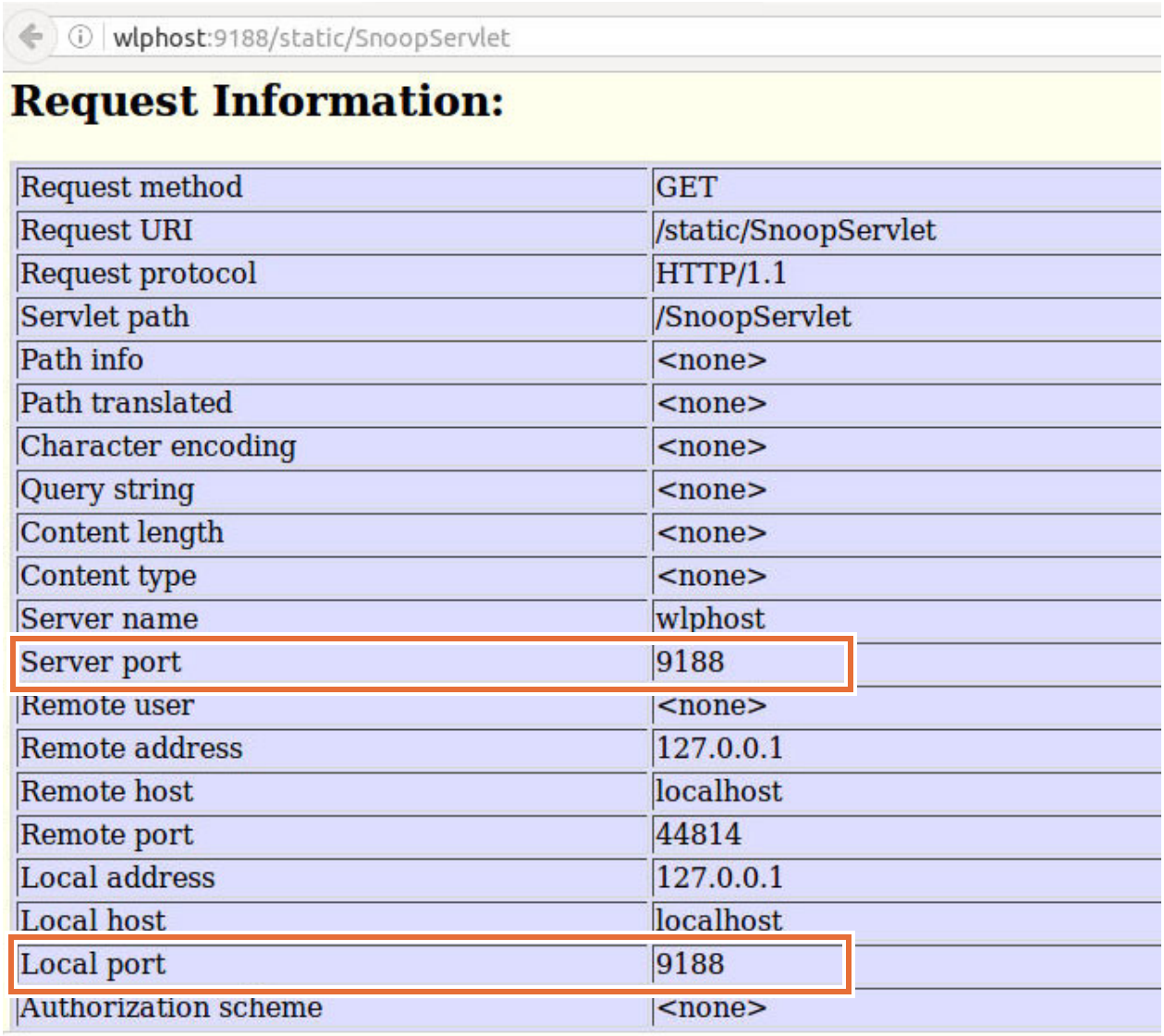
__ 4. Test the configuration.

__ a. Open a new tab in the Firefox browser window.

__ b. Access the static application directly in member static_1 with port 9188. Enter the following URL:

```
http://wlphost:9188/static/SnoopServlet
```

__ c. Go to the Request Information section. Notice that the Server port is 9188 and the Local port is 9188. This port is the port for the server static_1.



__ d. Access the application directly in member static_2 with port 9189. Enter the following URL:

`http://wlphost:9189/static/SnoopServlet`

__ e. Go to the Request Information section. Notice that the Server port is 9189 and the Local port is 9189. This port is the port for the server static_2.

__ f. Close the **SnoopServlet** tab.

__ g. Minimize the browser window.

Part 3: Configuring the IBM HTTP Server and plug-in

The IBM HTTP Server is already installed on the course image. By default, the IBM HTTP Server listens on port 80. In this part of the exercise, you change the default settings for the IBM HTTP Server.

__ 1. Verify the details in the `httpd.conf` file.

__ a. Go to the terminal window.

__ b. Go to `/opt/IBM/HTTPServer/conf` directory.

__ c. Open the `httpd.conf` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. In the `httpd.conf` file, verify or change the following details:

- Verify that Listen is set to `9180`

```
# Listen: Allows you to bind the web server to specific IP addresses
# and/or ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent the web server from accepting connections on all interfaces
# (0.0.0.0)
#
# Change this to "Listen 0.0.0.0:port" to restrict the server to
# IPv4.
#
#Listen 12.34.56.78:80
Listen 9180
```

- Change the last line in the file to:

`WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/IHS1/plugin-cfg.xml`

```
#LoadModule was_ap24_module /opt/IBM/WebSphere/Plugins/bin/64bits/
mod_was_ap24_http.so
#WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/IHS1/plugin-
cfg.xml

LoadModule was_ap24_module /opt/IBM/WebSphere/Plugins/bin/64bits/
mod_was_ap24_http.so
WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/IHS1/plugin-
cfg.xml
```

Reminder

Recall that the `bootstrap.properties` file for both static_1 and static_2 defined the `webServerPort` as 9180.

__ e. Save the changes.

__ f. Close the file when completed.

__ 2. Create a backup of the `plugin-cfg.xml` file.

__ a. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ b. Make a backup of the `plugin-cfg.xml` file. To make a backup, enter the following command:

```
cp plugin-cfg.xml plugin-cfg-backup.xml
```

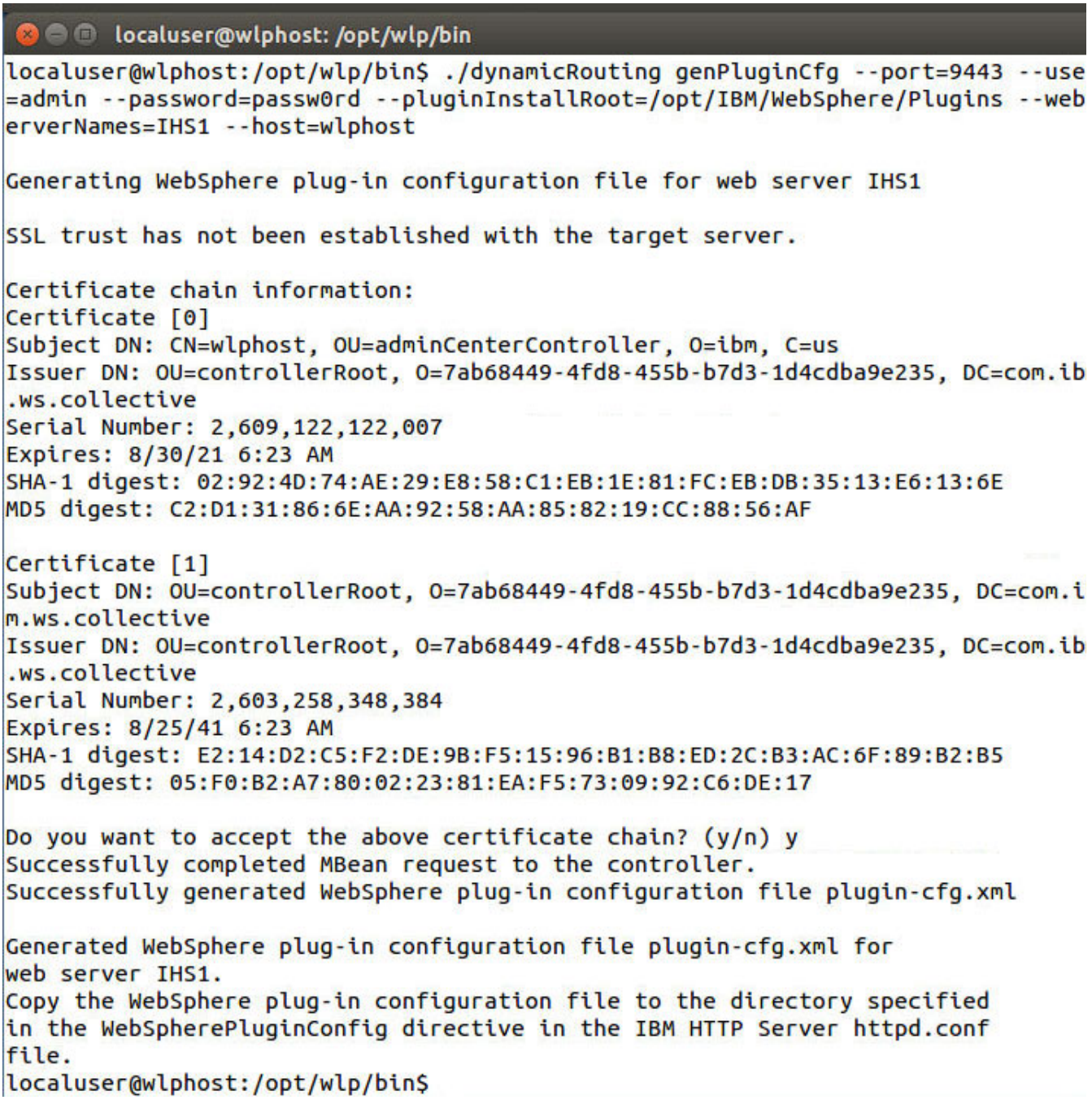
__ 3. For dynamic routing, a plug-in configuration file, and a secure connection are required between the plug-in and the collective controller. Next, configure the secure connection.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To generate the plug-in, enter the following command:

```
./dynamicRouting genPluginCfg --port=9443 --user=admin --password=passw0rd
--pluginInstallRoot=/opt/IBM/WebSphere/Plugins --webServerNames=IHS1
--host=wlphost
```

When prompted, accept the certificate chain.

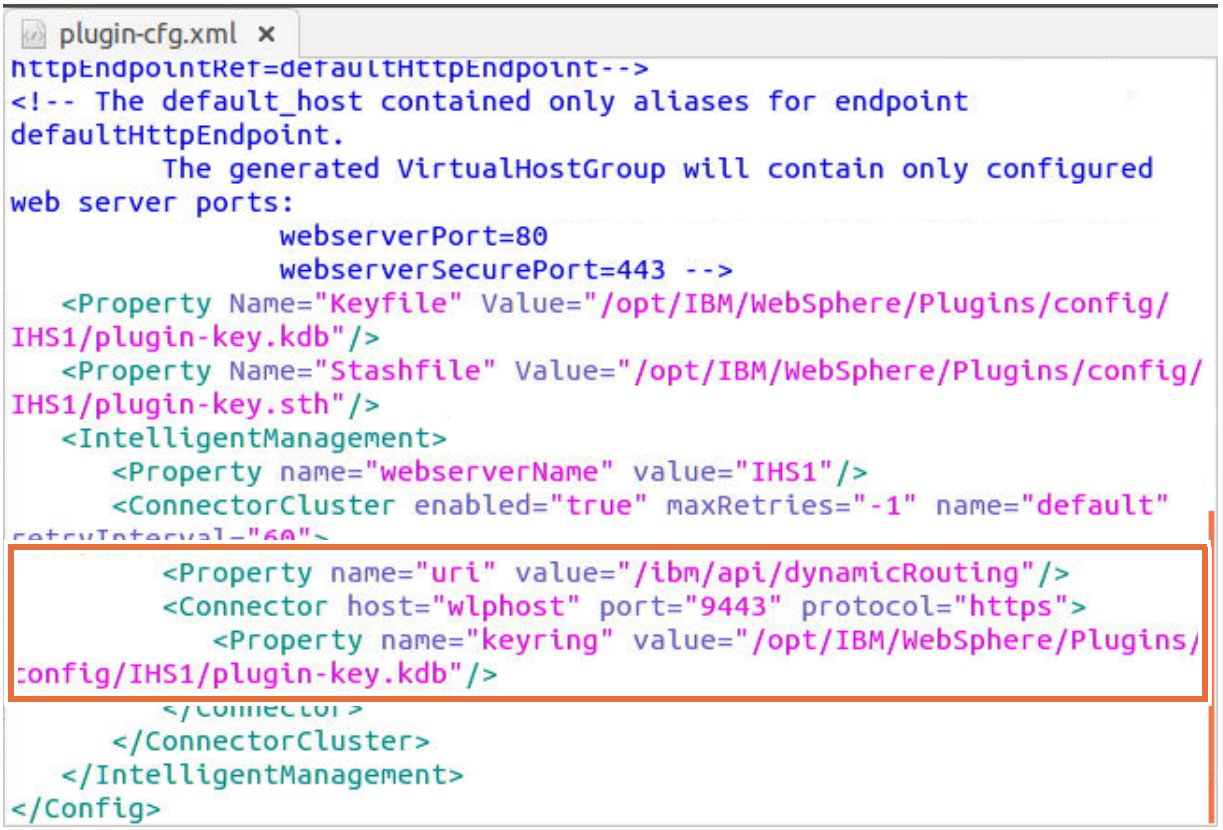


__ 4. Examine the generated `plugin-cfg.xml` file.

__ a. List the contents of the `/opt/wlp/bin` directory. You can see that the generated file is placed in this directory.

__ b. Open the `plugin-cfg.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ c. Go to the bottom of the file. You can see that the plug-in only needs to connect to the collective controller at port 9443 to get topology information. It does not need to know the host and port of the application servers. Also, note the location of the keystore file to enable security.



__ d. Close the file when completed.

__ 5. Copy the `plugin-cfg.xml` file to the correct location.

__ a. You must copy the plug-in file to the location that is defined in the `httpd.conf` file. This is the location where the plug-in is looking for the file. To copy the file, enter the following command:

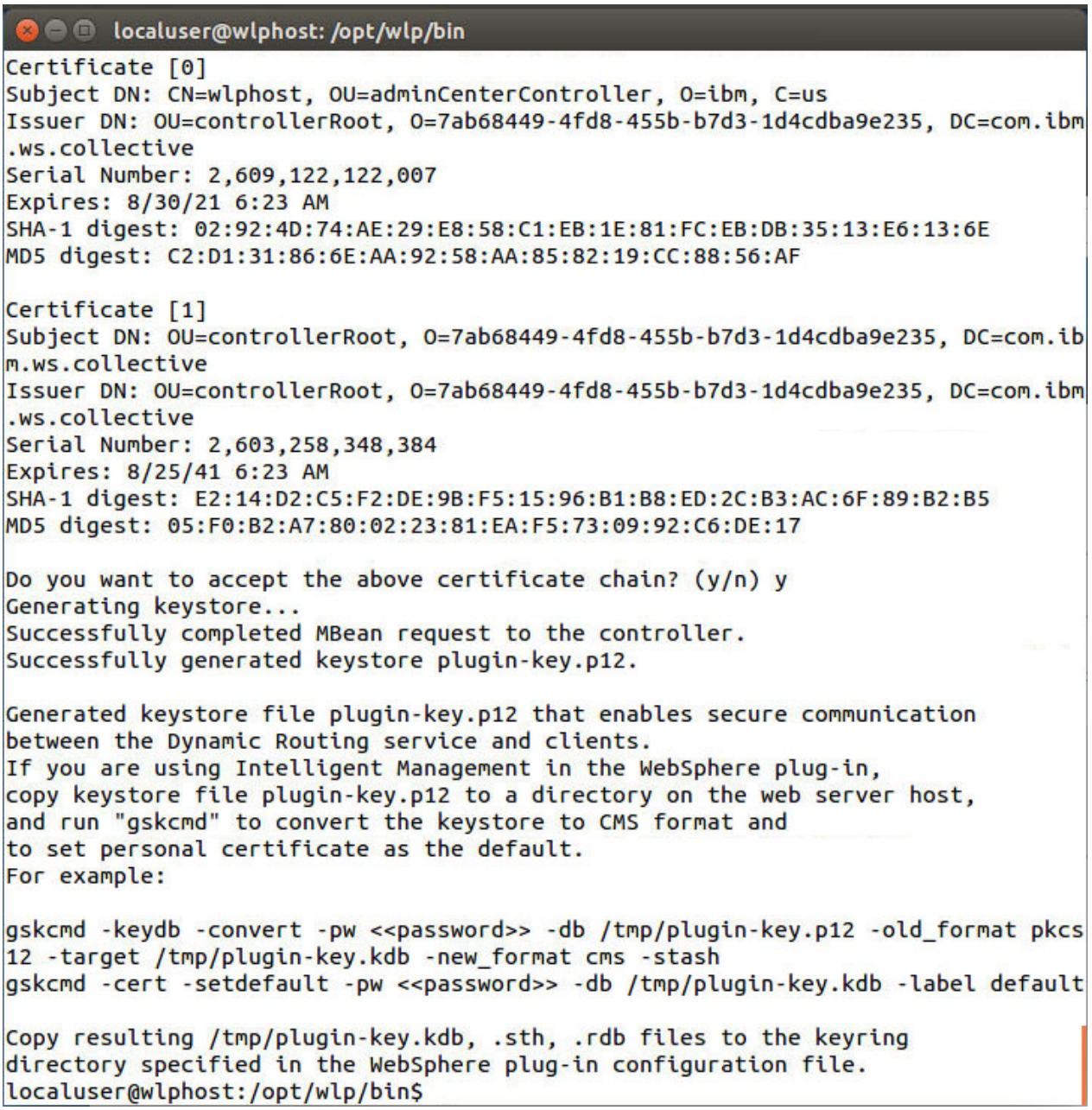
```
cp plugin-cfg.xml /opt/IBM/WebSphere/Plugins/config/IHS1/
```

__ 6. Generate the plug-in keystore.

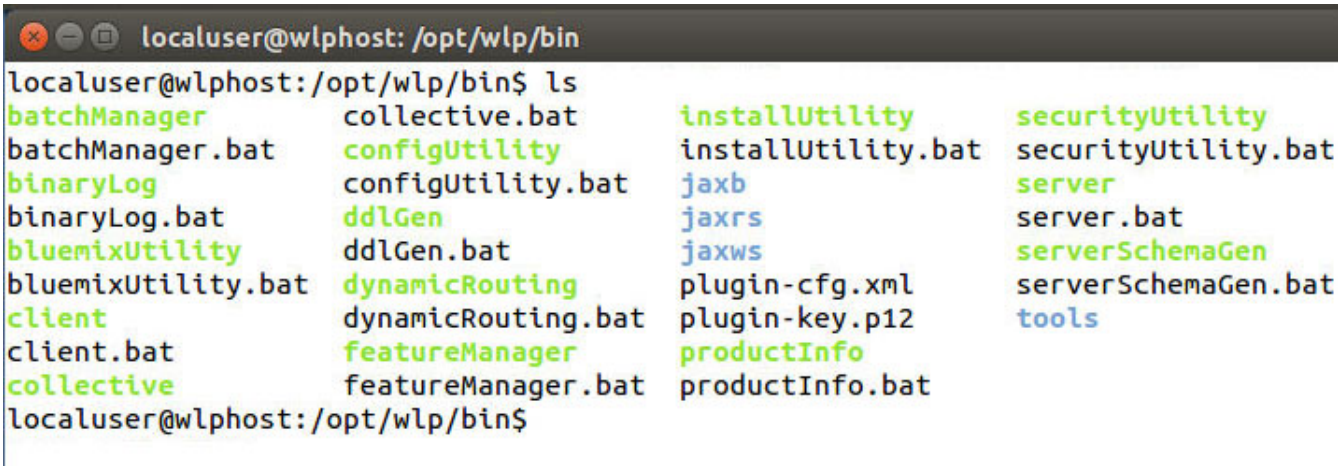
__ a. To generate the plug-in keystore, enter the following command:

```
./dynamicRouting genKeystore --port=9443 --user=admin --password=passw0rd
--keystorePassword=passw0rd --keystoreType=pkcs12 --host=wlphost
```

When prompted, accept the certificate chain.



__ b. List the contents of the directory. Notice that the output of the command is the `plugin-key.p12` file. However, the format of the certificate must be changed for the plug-in.



__ c. Create a temporary directory for the plug-in keystore and certification information. To create the directory, enter the following command:

```
mkdir /tmp/plugin
```

__ d. Copy the `plugin-key.p12` file to the temporary plug-in directory. To copy, enter the following command:

```
cp plugin-key.p12 /tmp/plugin/
```

__ e. List the contents of the `/tmp/plugin` directory.

__ 7. Convert the plug-in keystore from PKCS12 to CMS format, and update the default certificate in the keystore.

__ a. Update the `PATH` environment variable. To update variable, enter the following command:

```
export PATH=$PATH:/opt/IBM/HTTPServer/gsk8/lib64
```

__ b. Verify the `PATH` environment variable by using the `echo` command.

```
localuser@wlphost: /opt/wlp/bin
localuser@wlphost:/opt/wlp/bin$ export PATH=$PATH:/opt/IBM/HTTPServer/gsk8/lib64
localuser@wlphost:/opt/wlp/bin$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/opt/wlp/java/jre/bin:/opt/wlp/jython2.5.3/bin:/opt/IBM/HTTPServer/gsk8
/lib64
localuser@wlphost:/opt/wlp/bin$
```

__ c. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ d. Convert to CMS by entering the following command:

```
./gskcapicmd -keydb -convert -pw passw0rd -db /tmp/plugin/plugin-key.p12
-old_format pkcs12 -target /tmp/plugin/plugin-key.kdb -new_format cms -stash
-expire 365
```

Information

The Java command line interface to IKEYMAN, `gskcapicmd`, provides the necessary options to create and manage keys, certificates, and certificate requests. The native utility is always preferred over `gskcmd`. `gskcapicmd` is faster and some features are added to `gskcapicmd` before `gskcmd`.

__ e. List the contents of the plug-in directory by entering the following command:

```
ls /tmp/plugin
```

```
localuser@wlphost: /opt/IBM/HTTPServer/bin
localuser@wlphost:/opt/IBM/HTTPServer/bin$ ./gskcapicmd -keydb -convert -pw pass
w0rd -db /tmp/plugin/plugin-key.p12 -old_format pkcs12 -target /tmp/plugin/plugi
n-key.kdb -new_format cms -stash -expire 365
localuser@wlphost:/opt/IBM/HTTPServer/bin$ ls /tmp/plugin/
plugin-key.crl  plugin-key.kdb  plugin-key.p12  plugin-key.rdb  plugin-key.sth
localuser@wlphost:/opt/IBM/HTTPServer/bin$
```

__ f. Next, set the certificate as the default certificate. To make it the default, enter the following command:

```
./gskcapicmd -cert -setdefault -pw passw0rd -db /tmp/plugin/plugin-key.kdb
-label default
```

__ g. Copy the certificate files to the plug-in directory. To copy, enter the following command:

```
cp -r /tmp/plugin/* /opt/IBM/WebSphere/Plugins/config/IHS1/
```

__ h. List the contents of the plug-in directory. To list the contents, enter the following command:

```
ls /opt/IBM/WebSphere/Plugins/config/IHS1
```

```
localuser@wlphost: /opt/IBM/HTTPServer/bin
localuser@wlphost:/opt/IBM/HTTPServer/bin$ ./gskcapicmd -cert -setdefault -pw pa
ssw0rd -db /tmp/plugin/plugin-key.kdb -label default
localuser@wlphost:/opt/IBM/HTTPServer/bin$ cp -r /tmp/plugin/* /opt/IBM/WebSpher
e/Plugins/config/IHS1/
localuser@wlphost:/opt/IBM/HTTPServer/bin$ ls /opt/IBM/WebSphere/Plugins/config/
IHS1/
IHS1.responseFile      plugin-key.crl  plugin-key.rdb
plugin-cfg-backup.xml  plugin-key.kdb  plugin-key.sth
plugin-cfg.xml         plugin-key.p12  snoop2Cluster-plugin-cfg.xml
localuser@wlphost:/opt/IBM/HTTPServer/bin$
```

You can see that the directory contents are updated with the new files.

__ 8. Start the IBM HTTP Server

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. To start the HTTP Server, enter the following command:

```
./apachectl start
```

__ c. To see whether the HTTP Server is started, enter the following command:

```
ps -ef | grep httpd
```

```
localuser@wlphost: /opt/IBM/HTTPServer/bin
localuser@wlphost:/opt/IBM/HTTPServer/bin$ ps -ef | grep httpd
localus+  8816  1672  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd -
 /opt/IBM/HTTPServer -k start
localus+  8818  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd -
 /opt/IBM/HTTPServer -k start
localus+  8819  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd -
 /opt/IBM/HTTPServer -k start
localus+  8820  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd -
 /opt/IBM/HTTPServer -k start
localus+  8927 18692  0 09:08 pts/26   00:00:00 grep --color=auto httpd
localus+ 19975 18692  0 Sep02 pts/26   00:00:09 gedit httpd.conf
localuser@wlphost:/opt/IBM/HTTPServer/bin$
```

You can see that a number of processes are started.

Part 4: Testing the configuration

__ 1. Test the dynamic routing configuration.

__ a. Maximize the browser window.

__ b. Open a new tab in the Firefox browser.

__ c. Verify that the application works and test the configuration. Send the request to the web server by using the port, 9180, that you defined in the `httpd.conf` file. To test the configuration, enter the following URL:

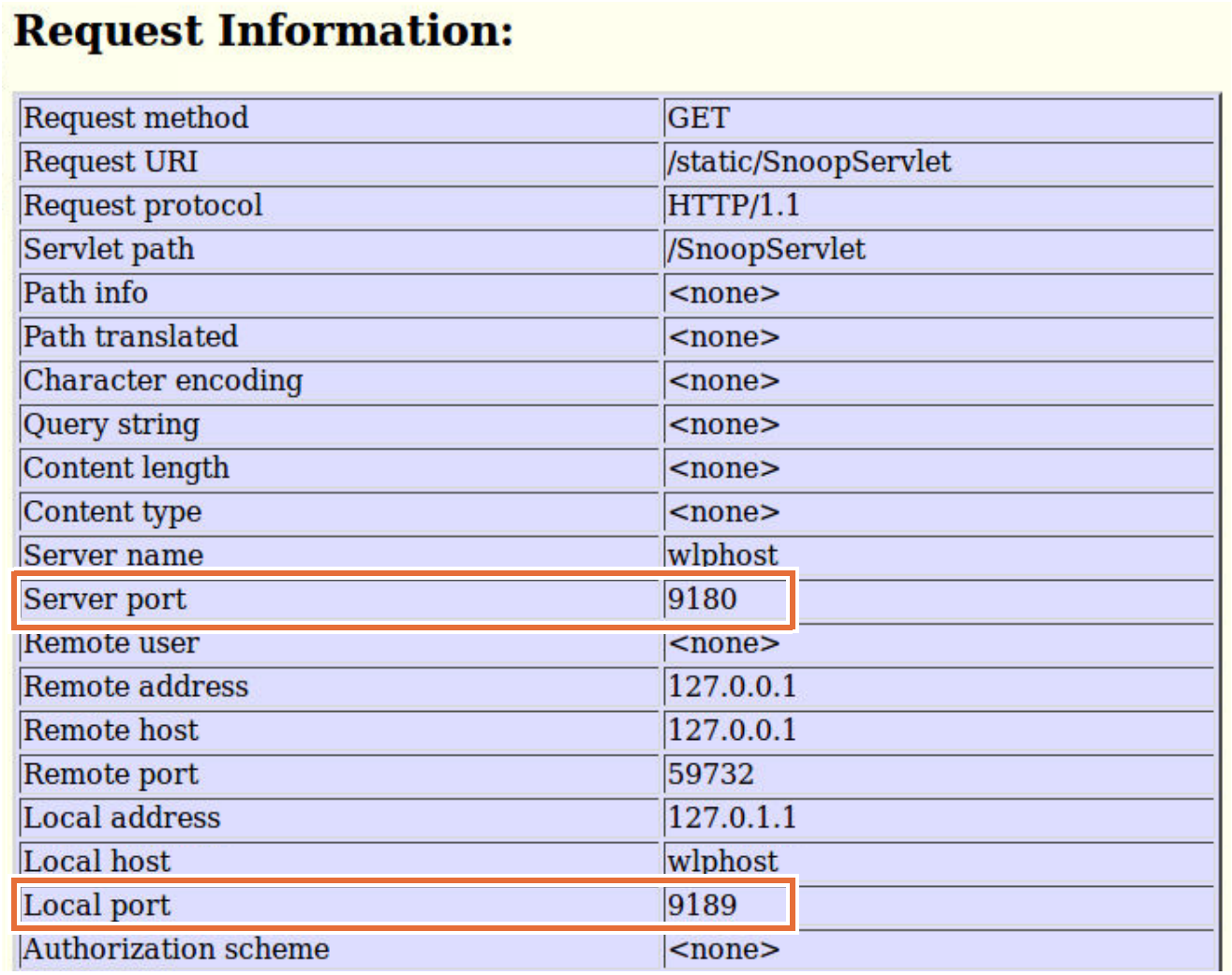
`http://wlphost:9180/static/SnoopServlet`

__ 2. Go to the Request Information section. Notice that the Server port is 9180, the web server, and the Local port is either 9188 or 9189. In this example, the request is sent to static_1 with port 9188.

Request Information:

Request method	GET
Request URI	/static/SnoopServlet
Request protocol	HTTP/1.1
Servlet path	/SnoopServlet
Path info	<none>
Path translated	<none>
Character encoding	<none>
Query string	<none>
Content length	<none>
Content type	<none>
Server name	wlphost
Server port	9180
Remote user	<none>
Remote address	127.0.0.1
Remote host	127.0.0.1
Remote port	38756
Local address	127.0.1.1
Local host	wlphost
Local port	9188
Authorization scheme	<none>

__ 3. Reload the page a few times, and verify that the Server port is 9180, the web server, and the Local port for the server is used. In this example, the request is sent to static_2 with port 9189.



Request Information:

Request method	GET
Request URI	/static/SnoopServlet
Request protocol	HTTP/1.1
Servlet path	/SnoopServlet
Path info	<none>
Path translated	<none>
Character encoding	<none>
Query string	<none>
Content length	<none>
Content type	<none>
Server name	wlphost
Server port	9180
Remote user	<none>
Remote address	127.0.0.1
Remote host	127.0.0.1
Remote port	59732
Local address	127.0.1.1
Local host	wlphost
Local port	9189
Authorization scheme	<none>

Your results might differ from the screen capture. In your test, the request might be sent to static_1 with port 9188.

__ d. Refresh the browser a few times to see how the requests are distributed to the servers.

__ 4. Stop server static_1.

__ a. Switch to the **Admin Center** tab.

__ b. Click the **Toolbox** icon**.**

__ c. Click **Explore**.

__ d. Click the **SERVERS** pane.

__ e. Click the **Actions** icon (down-arrow) in the upper right of the static_1 server box.

__ f. Click **Stop**.

__ g. In the stop window, click **Stop**. Wait for the status to change to Stopped.

__ 5. Test the configuration.

__ a. Switch to the **SnoopServlet** test tab.

__ b. Reload the page a few times. Verify that the requests are being sent to static_2 with port 9189.

__ 6. Stop server static_2.

__ a. Switch to the **Admin Center** tab.

__ b. First, start static_1. Click the **Actions** icon (down-arrow) in the upper right of the static_1 server box.

__ c. Click **Start**.

__ d. In the start window, click **Start**. Wait for the status to change to Running.

__ e. Click the **Actions** icon (down-arrow) in the upper right of the static_2 server box.

__ f. Click **Stop**.

__ g. In the stop window, click **Stop**. Wait for the status to change to Stopped.

__ 7. Test the configuration.

__ a. Switch to the **SnoopServlet** test tab.

__ b. Reload the page a few times. Verify that the requests are being sent to static_1 with port 9188.

__ 8. Stop both servers, static_1 and static_2.

__ a. Switch to the **Admin Center** tab.

__ b. Click the **Actions** icon (down-arrow) in the upper right of the static_1 server box.

__ c. Click **Stop**.

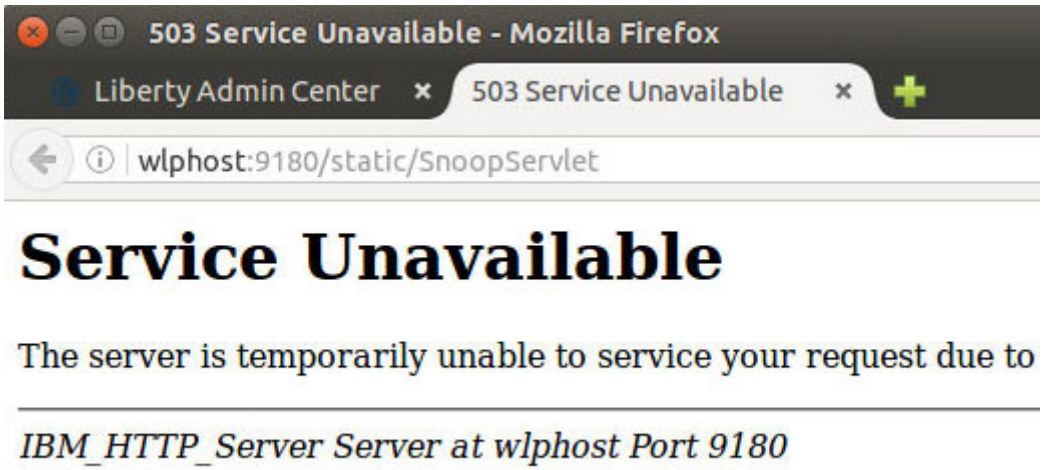
__ d. In the stop window, click **Stop**. Wait for the status to change to Stopped.

__ e. Verify that both static_1 and static_2 are stopped.

__ 9. Test the configuration.

__ a. Switch to the **SnoopServlet test** tab.

__ b. Reload the page. The request fails as there are no servers that are running.



__ 10. Start both servers, static_1 and static_2.

__ a. Switch to the **Admin Center** tab.

__ b. Click the **Actions** icon (down-arrow) in the upper right of the static_1 server box.

__ c. Click **Start**.

__ d. In the start window, click **Start**. Wait for the status to change to Running.

__ e. Click the **Actions** icon (down-arrow) in the upper right of the static_2 server box.

__ f. Click **Start**.

__ g. In the start window, click **Start**. Wait for the status to change to Running.

__ 11. Test the configuration.

__ a. Switch to the **SnoopServlet** test tab.

__ b. Reload the page a few times. Verify that the requests are being sent to each of the servers.

__ c. When completed, close the **SnoopServlet** tab.

Part 5: Cleaning up the environment

In this part of the exercise, you clean up the deployed artifacts.

__ 1. Stop the cluster.

__ a. Go to the Admin Center.

__ b. Click the **Dashboard** icon.

__ c. Click the **CLUSTERS** pane.

__ d. Click the **Actions** icon (down-arrow) in the upper right of the staticCluster box.

__ e. Click **Stop**.

__ f. In the stop window, click **Stop**. Wait for the status to change to Stopped.

__ g. Logout of the Admin Center.

__ h. Close the browser window.

__ 2. Stop the IBM HTTP Server

__ a. Go to the terminal window.

__ b. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ c. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ 3. Undeploy the servers.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. Set the `WLP_USER_DIR` environment variable as the servers are in a non-default directory. To set the variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/packagedServers/static_1/wlp/usr
```

__ c. Remove the cluster members from the collective controller. To remove static_1, enter the following command:

```
./collective remove static_1 --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ d. Set the `WLP_USER_DIR` environment variable as the servers are in a non-default directory. To set the variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/packagedServers/static_2/wlp/usr
```

__ e. To remove static_2, enter the following command:

```
./collective remove static_2 --host=wlphost --port=9443 --user=admin
--password=passw0rd --hostName=wlphost
```

When prompted, accept the chain certificate.

__ 4. Remove the directories and update an environment variable.

__ a. Go to the `/opt/wlp/packagedServers/` directory.

__ b. Remove the `static_1` and `static_2` directories.

__ c. Change the `WLP_USER_DIR` environment variable back to the default directory. To change the environment variable, enter the following command:

```
export WLP_USER_DIR=/opt/wlp/usr
```

__ 5. Stop the server adminCenterController.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop adminCenterController
```

__ c. Exit the terminal window.

End of exercise

Exercise review and wrap-up

In this exercise, you used the dynamic routing feature of Liberty to enable routing of HTTP requests to collective members.

Exercise 4. Auto-scaling

Estimated time

01:15

Overview

In this exercise, you learn how to enable the autonomic scaling capability of Liberty servers.

Objectives

After completing this exercise, you should be able to:

- Enable auto-scaling
- Configure and modify a scaling policy
- Package and deploy a dynamic cluster
- Test the auto-scaling feature

Introduction

Auto scaling provides an autonomic scaling capability of Liberty servers. The auto scaling functions are enabled by two Liberty features, scaling controller and scaling member. Auto scaling dynamically adjusts the number of Java virtual machines (JVMs) used to service your workload. This feature provides operational agility and decreases administrative tasks to enhance the resiliency of your middleware environment.

The conditions for auto scaling are defined by scaling policies. These conditions include the minimum or maximum number of server instances and the threshold values for each of the server resources.

Requirements

To complete this exercise, you need the WebSphere Liberty binary files and Liberty installed. You also need the collective controller, adminCenterController, created in the Managing Liberty collectives with the Admin Center exercise.

Exercise instructions

To set up an auto-scalable cluster, you need at least one collective controller with at least two member servers joined to the collective controller. Then, you need to configure the auto scaling features. Two features that provide the auto scaling capabilities for a Liberty cluster. The features include the following features:

- **scalingController-1.0**: The scaling controller feature decides when to expand or contract an auto scaling cluster. Collective controllers are required because they provide administration functions that uses the ability of the collective controller to manage the scaling controller. Only one of the running scaling controllers can make decisions.

  The scaling controller keeps the last known state of the scaling members. Stopping the scaling controller and changing the state of the scaling members might cause some invalid actions at the scaling controller restart until the scaling controller gathers the scaling member state.

- **scalingMember-1.0**:The scaling member feature monitors the workload within the server and its host, and when needed, it sends this information back to the scaling controller. The scaling member feature must be enabled as a collective member in a collective.

  All scaling member servers must also belong to a cluster because all policy information is applied at the cluster member.

Part 1: Configuring auto scaling in the collective controller

In this part of the exercise, you configure auto scaling and define scaling policies.

__ 1. Configure the scaling controller feature in the collective controller.

__ a. Open a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. Make a backup of the `server.xml` file. To make a copy, enter the following command:

```
cp server.xml server.backup
```

__ d. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

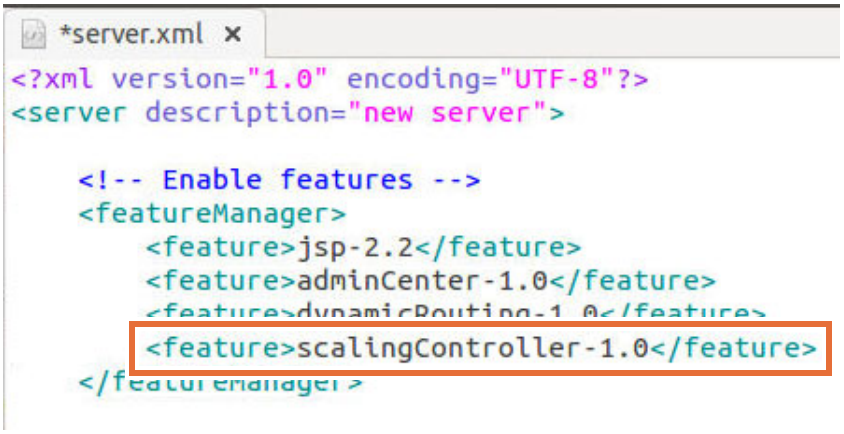
```
*server.xml ×
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>adminCenter-1.0</feature>
        <feature>dynamicRouting-1.0</feature>
    </featureManager>
```

You can see the configured features, which includes the `dynamicRouting-1.0` feature. Even though this exercise does not use dynamic routing to demonstrate auto

scaling, dynamic routing is required in a production environment to route requests correctly as the auto-scaling feature adjusts the number of running JVM instances.

__ e. Go to the feature manager section. Add the following feature:

```
<feature>scalingController-1.0</feature>
```



Information

Scaling policies give you the ability to control the scaling behaviors of clusters. Scaling policies are used by the scaling controller to determine when to start or stop members of a cluster. A cluster is scaled to meet a minimum or maximum number of servers per cluster or to meet resource demands of a cluster.

By default, a built-in scaling policy is embedded in the scaling controller. You can override the built-in scaling policy if needed by defining scaling policies in the `server.xml` file on the scaling controller. To override the built-in policy, use the `<defaultScalingPolicy>` element. The `<defaultScalingPolicy>` indicates the default policy to use for any auto-scaled cluster that does not have a specific scaling policy defined.

__ 2. Define the scaling policies to be used by the scaling controller.

__ a. Keep the `server.xml` file open. Next, you add the scaling policy.

__ b. Open another terminal window and go to the `/opt/labfiles/autoscale` directory.

__ c. Open the `policy1.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

*server.xml × policy1.xml ×

```
<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="true">
                <metric name="cpu" min="30" max="80"/>
                <metric name="heap" min="30" max="80"/>
                <metric name="memory" min="30" max="80"/>
                <in units="instance" amount="1" minInterval="30s"/>
                <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

This scaling policy defines the following details:

- If the host CPU or host memory exceeds 80% use, a new JVM instance is started on another host. This function is not demonstrated in the exercise as you use a single host only.
- If the instance heap exceeds 80% use, a new JVM instance is started on the same or different host, which depends on capacity.
- If host CPU, host memory, and one JVM heap fall below 30% use, the JVM instance is stopped.
- Policy decisions are made every 30 seconds.

__ d. Copy the entire scaling policy.

__ e. Go to the `server.xml` file that is still open.

__ f. Go to the bottom of the file before the `</featureManager>` tag.

__ g. Paste the scaling policy into the `server.xml` file for the controller.

*server.xml ×

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>adminCenter-1.0</feature>
        <feature>dynamicRouting-1.0</feature>
        <feature>scalingController-1.0</feature>
    </featureManager>

<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="true">
                <metric name="cpu" min="30" max="80"/>
                <metric name="heap" min="30" max="80"/>
                <metric name="memory" min="30" max="80"/>
                <in units="instance" amount="1" minInterval="30s"/>
                <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

Hint

You can also manually enter the following information into the file:

```
<scalingDefinitions>
      <defaultScalingPolicy min="1" max="4" enabled="true" >
            <metric name="cpu" min="30" max="80"/>
            <metric name="heap" min="30" max="80"/>
            <metric name="memory" min="30" max="80"/>
             <in units="instance" amount="1" minInterval="30s"/>
             <out amount="1" units="instance" minInterval="30s"/>
      </defaultScalingPolicy>
</scalingDefinitions>
```

__ h. Save and close the `server.xml` file.

__ i. Close the `policy1.xml` file.

__ 3. Start the collective controller.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To start the collective controller, enter the following command:

```
./server start adminCenterController
```

__ c. If needed, you can verify that the collective controller is started. To verify, enter the following command:

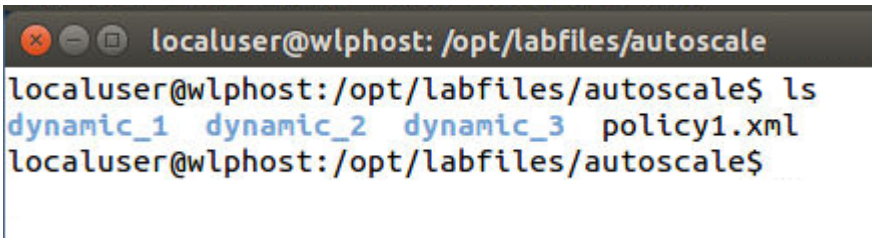
```
./server status adminCenterController
```

Part 2: Creating and deploying a dynamic cluster

__ 1. Examine the customized files for the server dynamic_1.

__ a. Go to the `/opt/labfiles/autoscale` directory.

__ b. List the contents of the directory. There is a directory for dynamic_1, dynamic_2, and dynamic_3. Each directory contains the customized files for a Liberty server runtime environment.



__ c. Go to the `dynamic_1` directory.

__ d. List the contents of the directory. You can see the customized files for the server and an apps directory that contains an application.

__ e. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ f. Go to the featureManager section. Notice that the file includes the following features:

- **collectiveMember-1.0**: This feature indicates that the controller manages the server.
- **clusterMember-1.0**: This feature indicates that the server is to be a member of a cluster.
- **scalingMember-1.0**: This feature monitors the workload within a server and its host, then sends this information to the scaling controller.
- **localConnector1.0**: This feature allows the use of a local JMX connector that is built into the JVM to access JMX resources in the server. This feature is not required for auto scaling.
- **restConnector-1.0**: This feature enables remote access by JMX clients by using a REST-based connector and requires SSL and basic user security configuration. This feature is required for the controller to push policy information down to the application servers.

server.xml

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>collectiveMember-1.0</feature>
        <feature>clusterMember-1.0</feature>
        <feature>scalingMember-1.0</feature>
        <feature>localConnector-1.0</feature>
        <feature>restConnector-1.0</feature>
    </featureManager>
```

__ g. Go to the bottom of the file. You can see the cluster, dynamicCluster, is defined. Note the element for hostSingleton. Each scaling member needs to define a hostSingleton element with a port in the `server.xml`. All scaling members on the same host must use the same port. You can specify any port number, but the port number must be unique on the host computer.

By specifying the same port number for all scaling members on the same host, a member is automatically elected as a singleton to gather metrics on the other members on the same host, to send to the scaling controller. If the member that is acting as the singleton is stopped, another member automatically takes over.

server.xml

```
allowFromEndpointRef="defaultHttpEndpoint" />

    <!-- use virtual host application_host for applications -->
    <virtualHost id="application_host"
allowFromEndpointRef="ApplicationHttpEndpoint" >
        <hostAlias>wlphost:${httpPort}</hostAlias>
        <hostAlias>wlphost:${httpsPort}</hostAlias>
        <hostAlias>${webServerHost}:${webServerPort}</hostAlias>
    </virtualHost>

   <webApplication id="dynamic" location="dynamic.war"  />

   <clusterMember name="dynamicCluster" />

   <hostSingleton name="ScalingMemberSingletonService"
port="${singletonServicePort}" />


   <logging
traceSpecification="*=info:Scaling=all:com.ibm.ws.imf.*=all:com.ibm.ws.
>

</server>
```

__ h. When completed, close the file.

__ i. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.

bootstrap.properties

```
httpPort=9191
httpsPort=9554
httpAdminPort=9091
httpsAdminPort=9454
webServerHost=*
webServerPort=9180
singletonServicePort=5164
```

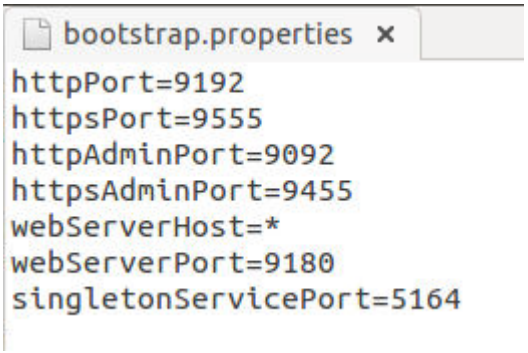
You can see the ports that are defined for this server. Note the httpPort is 9191 and the singletonServicePort is 5164.

__ j. When completed, close the file.

__ 2. Examine the customized files for the server dynamic_2.

__ a. Go to the `/opt/labfiles/autoscale/dynamic_2` directory and list the contents of the directory. You can see the same files, each customized for dynamic_2.

__ b. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.



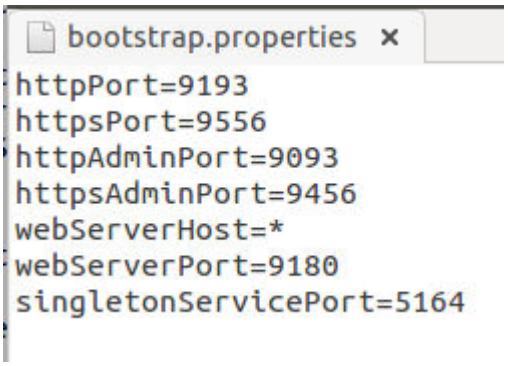
You can see the ports that are defined for this server. Note the httpPort is 9192 and the singletonServicePort is 5164.

__ c. When completed, close the file.

__ 3. Examine the customized files for the server dynamic_3.

__ a. Go to the `/opt/labfiles/autoscale/dynamic_3` directory and list the contents of the directory. You can see the same files, each customized for dynamic_3.

__ b. Open the `bootstrap.properties` file and examine the settings. Open the file by using an editor such as vi or gedit.



You can see the ports that are defined for this server. Note the httpPort is 9193 and the singletonServicePort is 5164.

__ c. When completed, close the file.

__ 4. Copy the `dynamic_1` directory to the `servers` directory.

__ a. Go to the `/opt/labfiles/autoscale` directory.

__ b. Copy the `dynamic_1` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

```
cp -r dynamic_1 /opt/wlp/usr/servers/
```

__ 5. Create a packaged server. From the command line, you can create a compressed file that contains a Liberty runtime environment, the files in the shared resources directory, a specific server, and the applications that are embedded in the server.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To create the packaged server `dynamic_1`, enter the following command:

```
./server package dynamic_1 --include=minify
```

Wait until you see that the package is complete. You can also see that this command creates a file, `dynamic_1.zip`, in the `dynamic_1` directory.

__ 6. Start the Admin Center console.

__ a. Open the Firefox web browser and go to the following website:

```
http://wlphost:9080/adminCenter
```

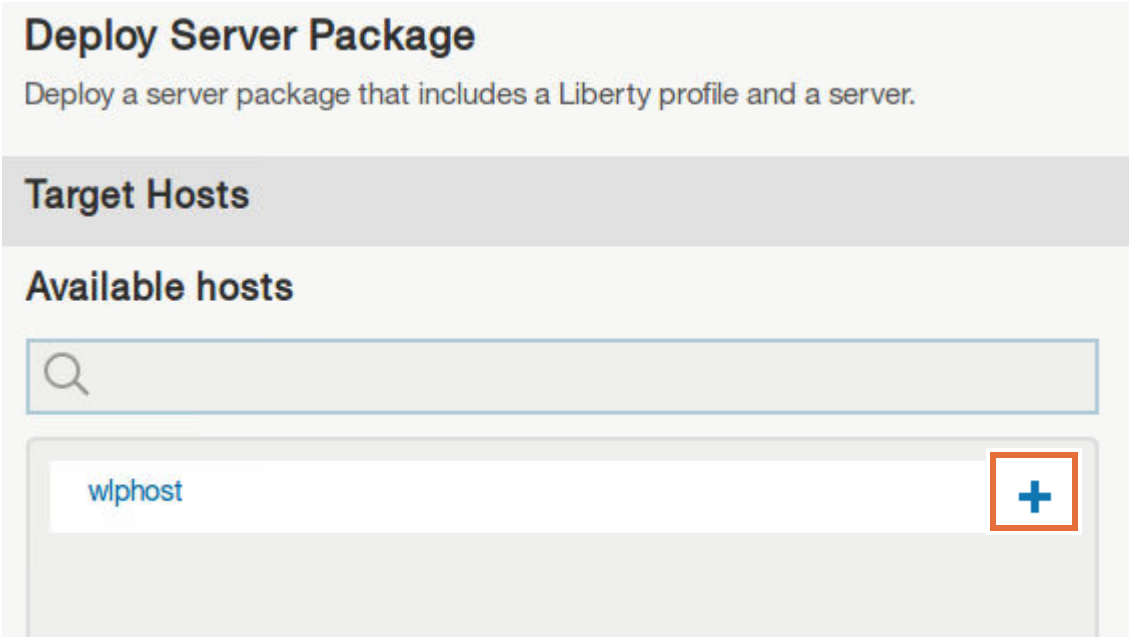
__ b. In the login area, enter `admin` as the User Name and `passw0rd` as the Password and click **Submit**.

__ 7. Deploy a packaged server by using the Admin Center.

__ a. In the Admin Center, click **Deploy**.

__ b. Go to the Available hosts section.

__ c. Select the host that you want to deploy the package to. Next to wlphost, click **+**.



You can see that the host, wlphost, is moved to the Selected hosts section.

__ d. Go to the Settings section.

__ e. Under Server Package, verify that the option **Upload a server package file** is selected.

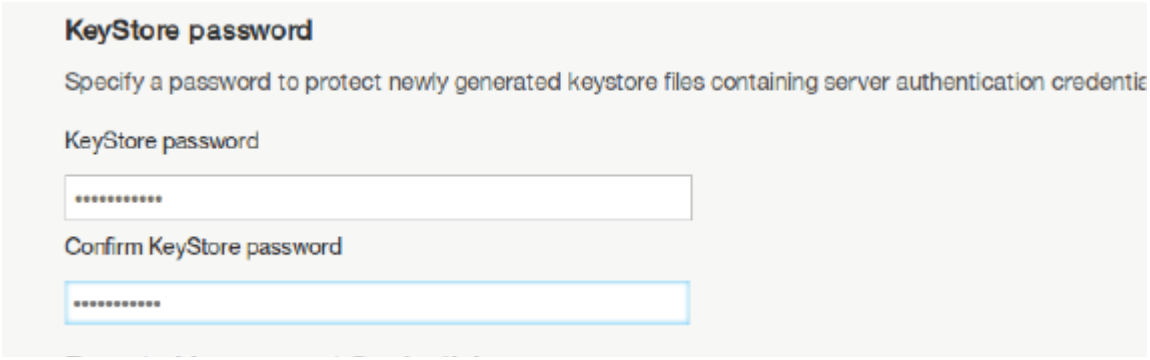
__ f. Click **Browse**, select `/opt/wlp/usr/servers/dynamic_1/dynamic_1.zip`, and click **Open**.

__ g. In the **Target Directory** field, enter the following directory:

```
/opt/wlp/packagedServers/dynamic_1
```

__ h. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`

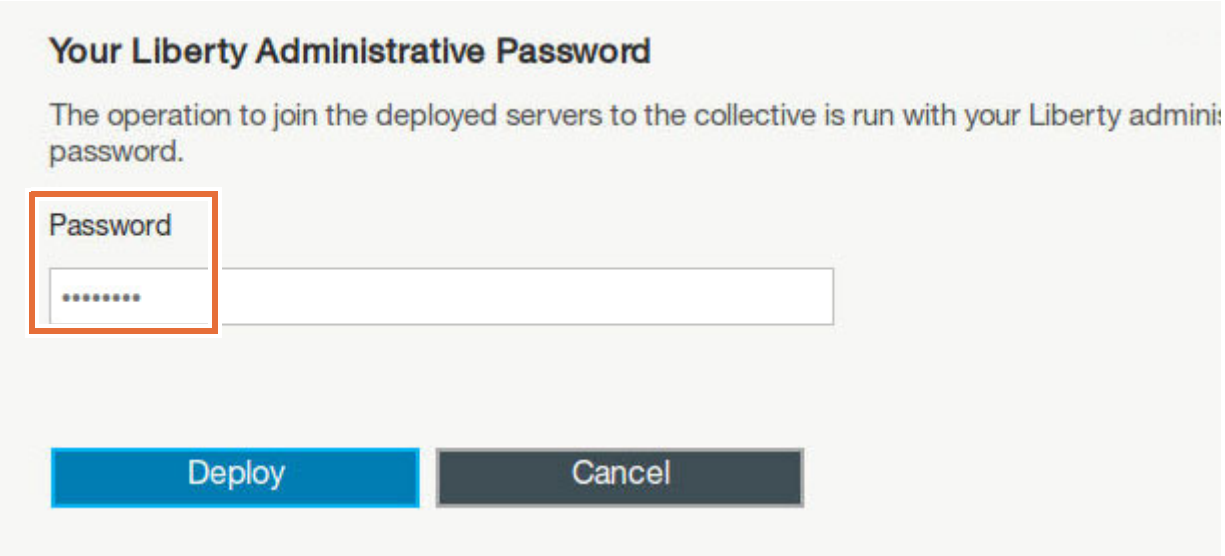


__ i. Go to the Remote Management Credentials section.

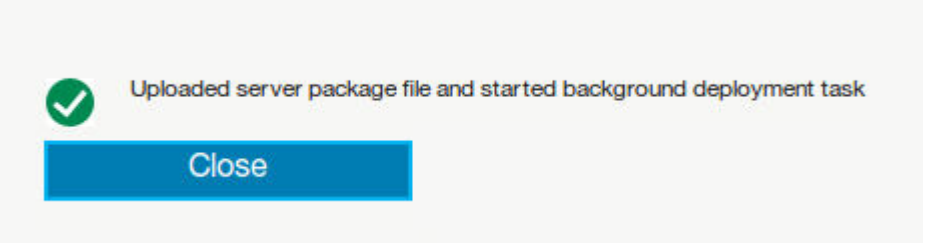
__ j. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

__ k. Go to the Your Liberty Administrative Password section.

__ l. In the **Password** field, enter: `passw0rd`

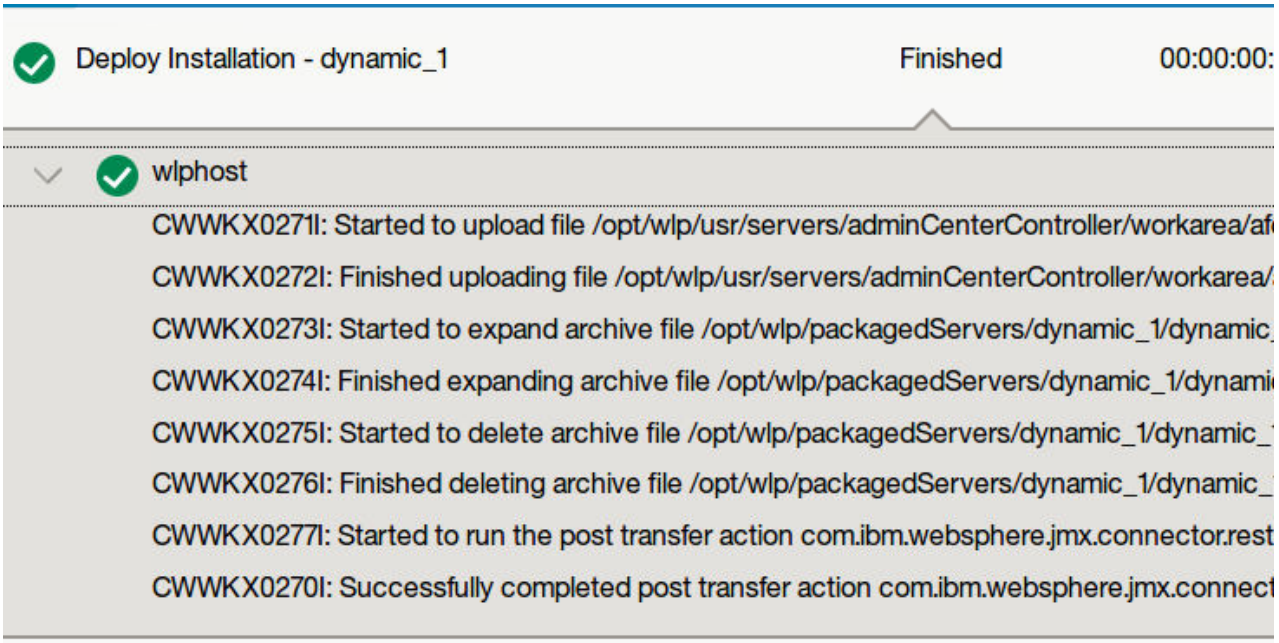


__ m. At the bottom of the page, click **Deploy.** Wait for the message that the file is uploaded and a background deployment task is started.



__ n. Click **Task Details and History** to see the status of the deployment.

__ o. The green check mark and green progress bar indicate a successful deployment. Click **Deploy Installation - dynamic_1 > wlphost** to view the details of the task.



__ p. Minimize the Admin Center browser.

__ 8. Examine what the collective controller deployed.

__ a. Go to the terminal window.

__ b. Go to the `/opt/wlp/packagedServers/dynamic_1/wlp/usr/servers/dynamic_1` directory.

__ c. List the contents of the directory. Notice that the collective controller creates the `collective-join-include.xml` file that allows the member to join with the controller.

__ d. Open the `collective-join-include.xml` file with an editor such as vi or gedit. Examine the settings in the file. You can see the details about the collective controller.

__ e. When completed, close the file.

__ 9. Repeat the prior steps to package and deploy the dynamic_2 packaged server.

__ a. Go to the `/opt/labfiles/autoscale` directory.

__ b. Copy the `dynamic_2` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

```
cp -r dynamic_2 /opt/wlp/usr/servers/
```

__ c. Go to the `/opt/wlp/bin` directory.

__ d. To create the packaged server `dynamic_2`, enter the following command:

```
./server package dynamic_2 --include=minify
```

__ e. Maximize the Admin Center.

__ f. In the Admin Center, click the **Toolbox** icon to go back to the toolbox.

__ g. Click **Deploy**.

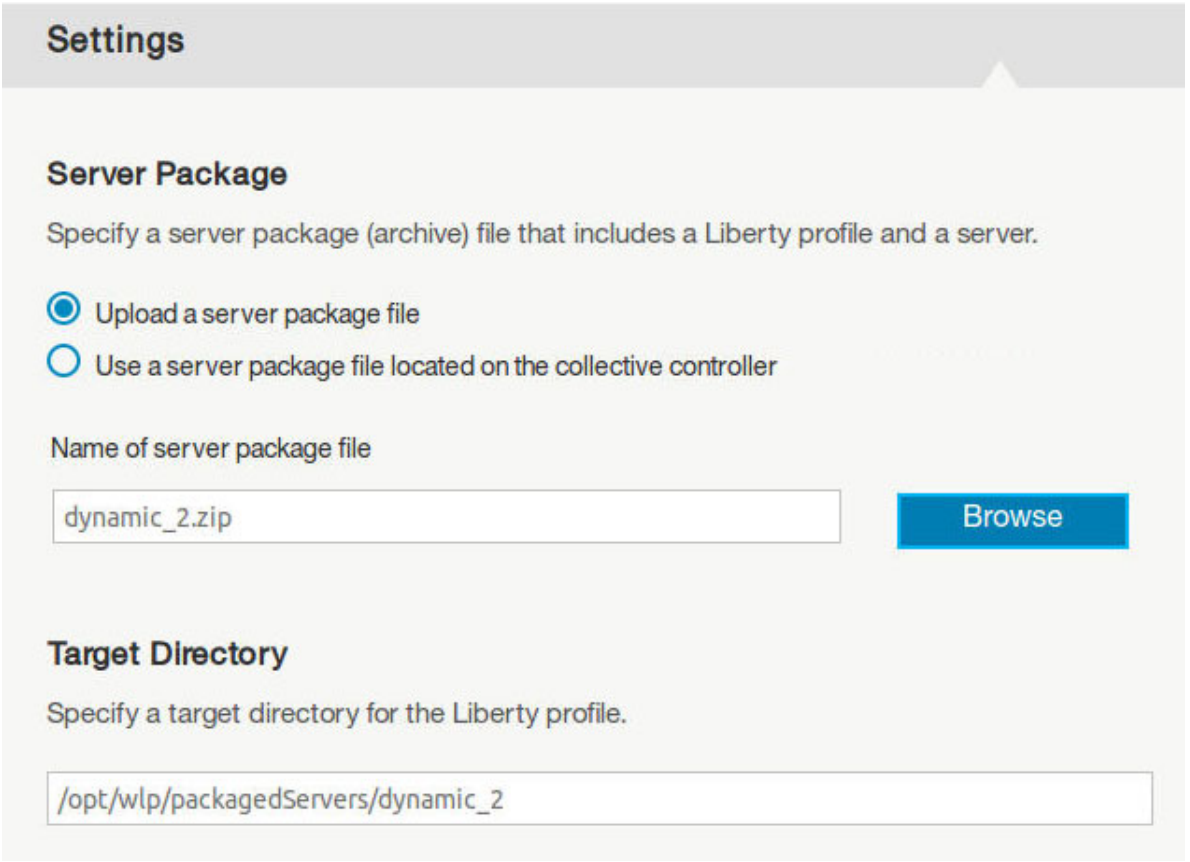
__ h. Go to the Available hosts section. Select the host that you want to deploy the package to. Next to wlphost, click **+**.

__ i. Go to the Settings section. Under Server Package, verify that the option **Upload a server package file** is selected.

__ j. Click **Browse,** select `/opt/wlp/usr/servers/dynamic_2/dynamic_2.zip,` and click **Open**.

__ k. In the **Target Directory** field, enter the following directory:

`/opt/wlp/packagedServers/dynamic_2`



__ l. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`

__ m. Go to the Remote Management Credentials section. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

__ n. Go to the Your Liberty Administrative Password section. In the **Password** field, enter: `passw0rd`

__ o. At the bottom of the page, click **Deploy.** Wait for the message the file is uploaded and a background deployment task is started.

__ p. To see the Task History, click the **Background Tasks** icon at the upper-right corner of the page.

__ q. Click **Task Details and History** to see the status of the deployment.

__ r. The green check mark and green progress bar indicate a successful deployment.

__ 10. Repeat the prior steps to package and deploy the dynamic_3 packaged server.

__ a. Go to the `/opt/labfiles/autoscale` directory.

__ b. Copy the `dynamic_3` directory to the `/opt/wlp/usr/servers/` directory. To copy, enter the following command:

```
cp -r dynamic_3 /opt/wlp/usr/servers/
```

__ c. Go to the `/opt/wlp/bin` directory.

__ d. To create the packaged server `dynamic_3`, enter the following command:

```
./server package dynamic_3 --include=minify
```

__ e. Maximize the Admin Center.

__ f. In the Admin Center, click the **Toolbox** icon to go back to the toolbox.

__ g. Click **Deploy**.

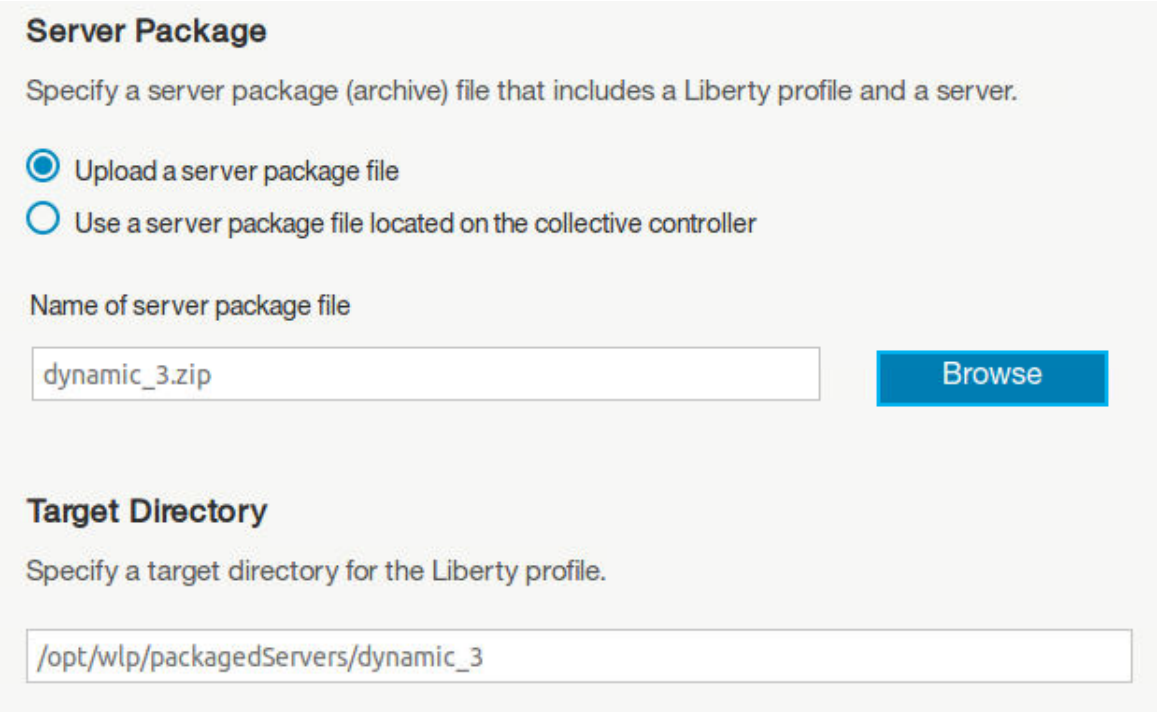
__ h. Go to the Available hosts section. Select the host that you want to deploy the package to. Next to wlphost, click **+**.

__ i. Go to the Settings section. Under Server Package, verify that the option **Upload a server package file** is selected.

__ j. Click **Browse,** select `/opt/wlp/usr/servers/dynamic_3/dynamic_3.zip,` and click **Open**.

__ k. In the **Target Directory** field, enter the following directory:

`/opt/wlp/packagedServers/dynamic_3`



__ l. Go to the KeyStore password section and enter the following details:

- For **KeyStore password**, enter: `passw0rd`
- For **Confirm KeyStore password**, enter: `passw0rd`

__ m. Go to the Remote Management Credentials section. Keep the default security setting, which is **Use the connection method and credentials configured for each target host**.

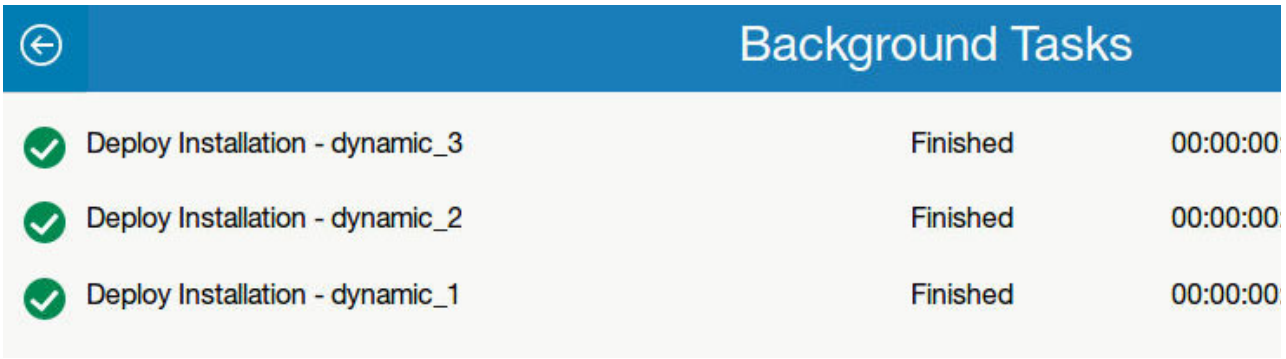
__ n. Go to the Your Liberty Administrative Password section. In the **Password** field, enter: `passw0rd`

__ o. At the bottom of the page, click **Deploy.** Wait for the message that the file is uploaded and a background deployment task is started.

__ p. To see the Task History, click the **Background Tasks** icon at the upper-right corner of the page.

__ q. Click **Task Details and History** to see the status of the deployment.

__ r. The green check mark and green progress bar indicate a successful deployment.



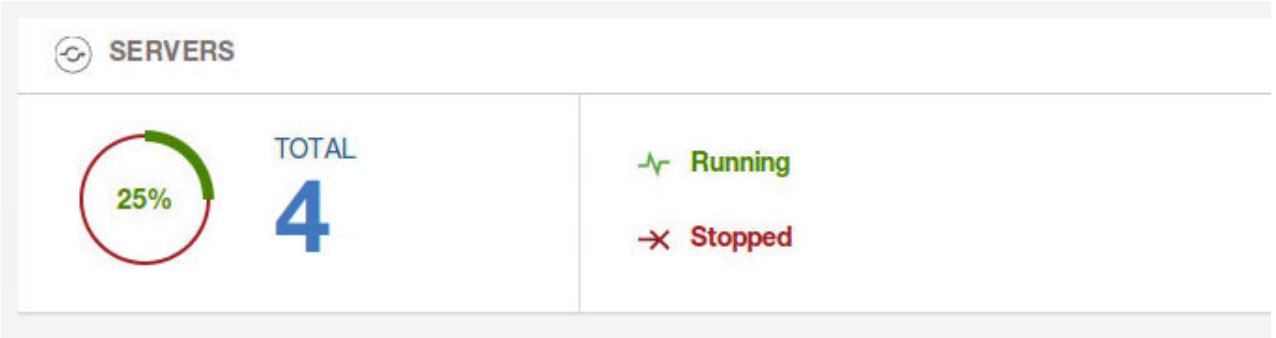
Part 3: Examining the dynamic cluster configuration

__ 1. Examine the configuration.

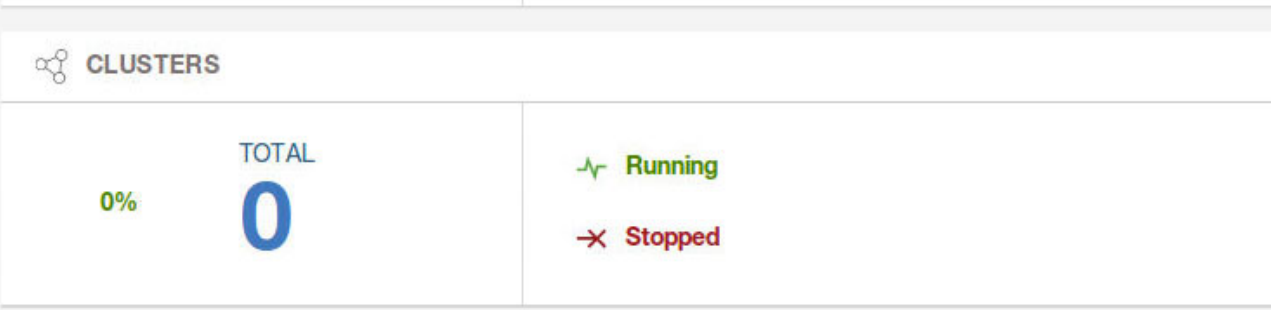
__ a. Go back to the Toolbox.

__ b. Click **Explore**.

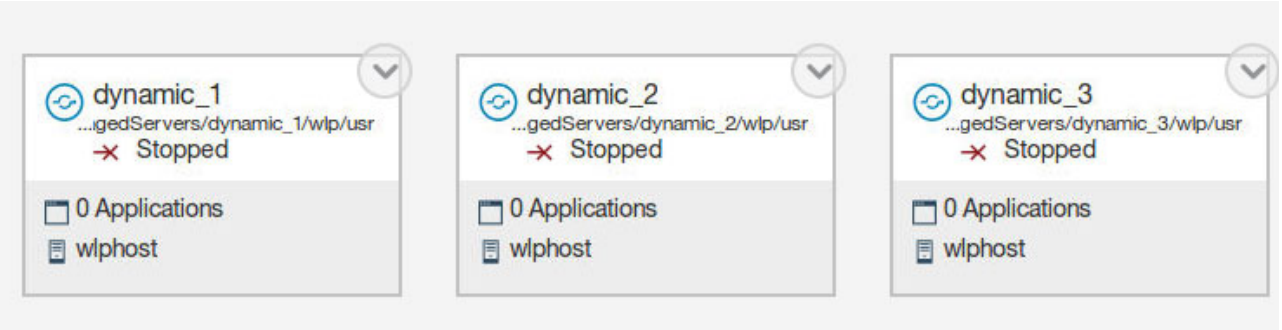
__ c. Go to the SERVERS section. There are now three new servers. The total is four, which includes the adminCenterController.



__ d. Go to the CLUSTERS section. Notice that no cluster is detected yet.



__ e. Click the **SERVERS** pane. You should see dynamic_1, dynamic_2 and dynamic_3 in Stopped state.



__ f. Minimize the Admin Center.

__ 2. Examine the log file for the controller.

Information

Keep in mind, that some of the servers might not be started or stopped because of a scaling policy breach. However, the breach might not occur on your machine depending on the hardware you have.

For the controller to stop the application server on your hardware based on the policy that is defined, you must have:

- Less than 30% heap that is being used by the server
- Less than 30% cpu use on the host
- Less than 30% total memory use on the host

To determine whether the controller has taken a policy action, look in the `messages.log` file of the adminCenterController.

__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController/logs` directory.

__ c. Tail the `messages.log` file for the collective controller to monitor the output. To tail the log file, enter the following command:

```
tail -f messages.log
```

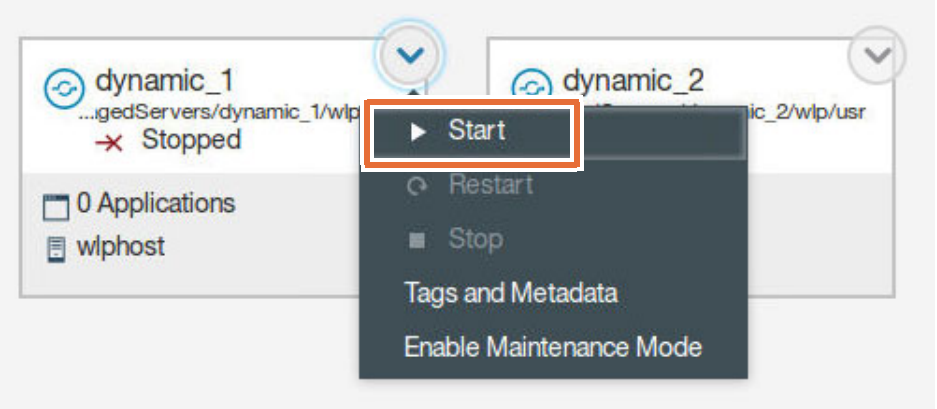
__ d. Keep the window open where the tail command is running so you can refer to it often.

__ 3. Start the servers.

__ a. Maximize the Admin Center.

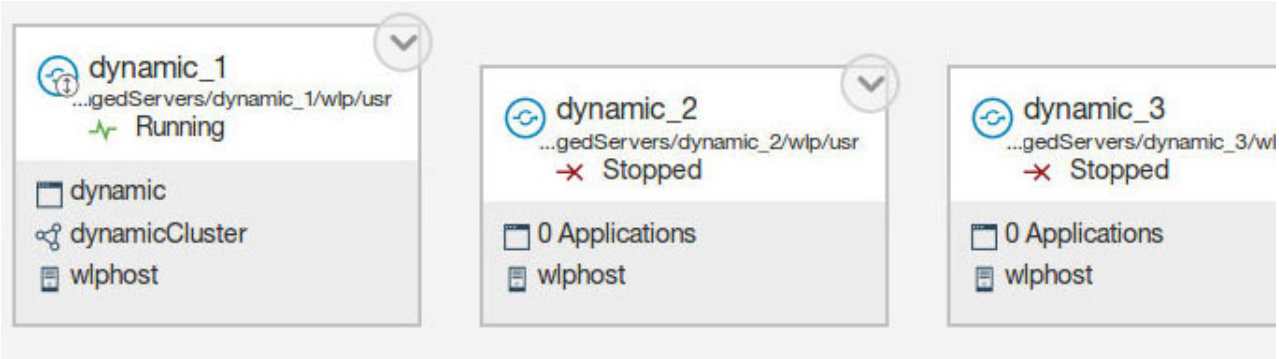
__ b. Click the **Actions** icon (down-arrow) in the upper right of the dynamic_1 server box.

__ c. Then, click **Start** to start the dynamic_1 server.



__ d. In the start server dynamic_1 window, click **Start**.

__ e. After a few minutes, the status of the server changes to **Running**. A cluster is now detected and the application is running.



__ a. Click the **Actions** icon (down-arrow) in the upper right of the dynamic_2 server box.

__ b. Then, click **Start** to start the dynamic_2 server.

__ c. In the start server dynamic_2 window, click **Start**.

__ d. After a few minutes, the status of the server changes to **Running**.

__ a. Click the **Actions** icon (down-arrow) in the upper right of the dynamic_3 server box.

__ b. Then, click **Start** to start the dynamic_3 server.

__ c. In the start server dynamic_3 window, click **Start**.

__ d. After a few minutes, the status of the server changes to **Running**.

__ e. Minimize the Admin Center browser.

__ f. Examine the terminal window where the tail command is running. The controller might start and stop various servers in the dynamic cluster based on the defined policy.

Information

If the controller has taken action, the `messages.log` entries look like the following entries:

```
[ ] 00000061 com.ibm.ws.scaling.controller.internal.ScalingExecutorImpl   I
CWWKV0115I: The scaling controller is stopping server dynamic_2 in user directory
/opt/wlp/packagedServers/dynamic_2/wlp/usr on host wlphost to reduce capacity in
cluster dynamicCluster.
[ ] 00000061 com.ibm.ws.scaling.controller.internal.ScalingExecutorImpl   I
CWWKV0114I: The scaling controller has successfully stopped server dynamic_2 on
host wlphost.
[ ] 0000002e com.ibm.ws.scaling.controller.topology.RepositoryMonitor     I
CWWKV0121I: The server dynamic_3 in user directory
/opt/wlp/packagedServers/dynamic_3/wlp/usr on host wlphost has been defined as a
scaling member in cluster dynamicCluster.
[ ] 0000002e com.ibm.ws.scaling.controller.topology.RepositoryMonitor     I
CWWKV0121I: The server dynamic_2 in user directory
/opt/wlp/packagedServers/dynamic_2/wlp/usr on host wlphost has been defined as a
scaling member in cluster dynamicCluster.
```

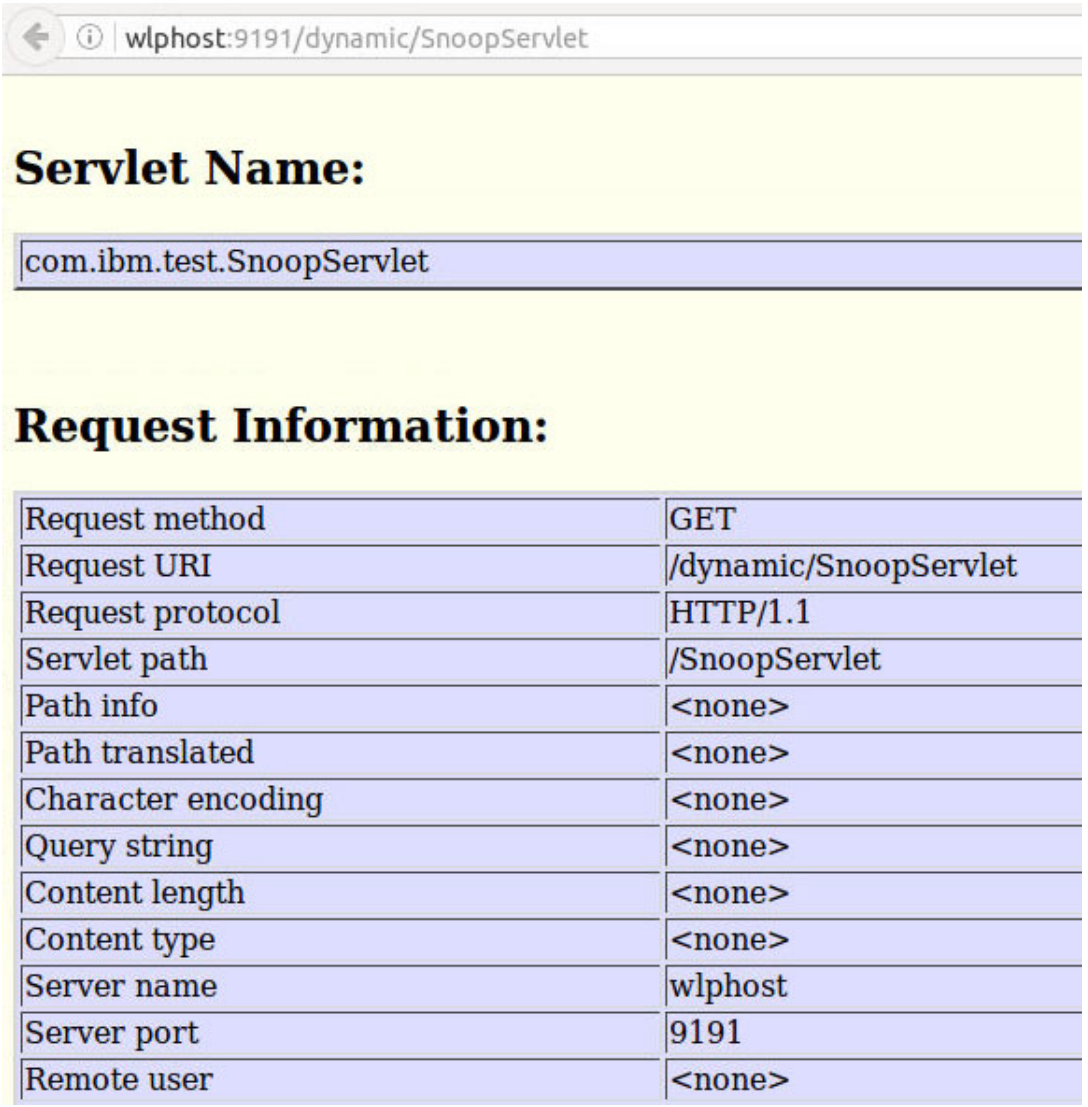
__ 4. Test the SnoopServlet on each server in the dynamic cluster.

__ a. Maximize the browser window.

__ b. Open a new tab in the browser.

__ c. To test the configuration on dynamic_1, enter the following URL:

`http://wlphost:9191/dynamic/SnoopServlet`



__ d. To test the configuration on dynamic_2, enter the following URL:

`http://wlphost:9192/dynamic/SnoopServlet`

__ e. To test the configuration on dynamic_3, enter the following URL:

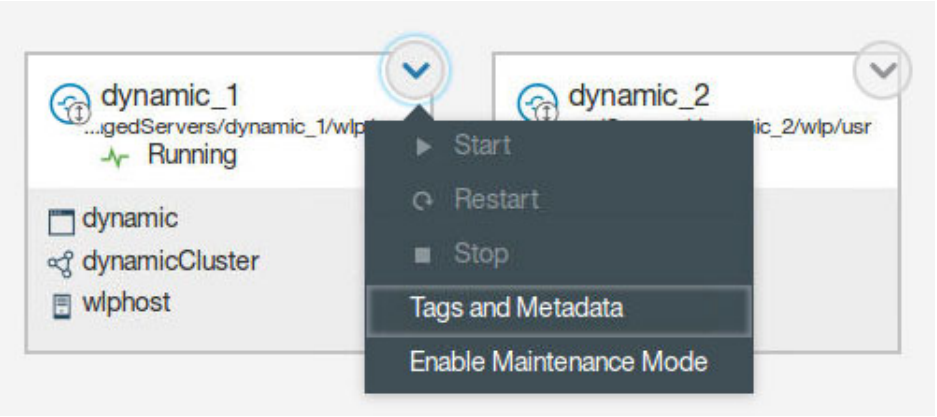
`http://wlphost:9193/dynamic/SnoopServlet`

__ 5. Examine the servers in the Admin Center.

__ a. Go to the **Admin Center** tab.

__ b. Verify that you are on the Servers pane.

__ a. Click the **Actions** icon (down-arrow) in the upper right of the dynamic_1 server box.



__ b. You are unable to start or stop the server in the dynamic cluster from the Admin Center. The reason is that you should define the scaling policy only and have the scaling controller adjust the number of JVMs.

__ c. Minimize the browser window.

Part 4: Testing the minimum number of servers

In this part of the exercise, you test whether the controller maintains a minimum number of servers to be running. To test, you stop all the servers in the dynamic cluster by using the command line. The controller might stop some of the servers if a policy breach was detected for your environment.

__ 1. Stop the servers in the dynamic cluster.

__ a. Open a terminal window.

__ b. Go to the `/opt/wlp/packagedServers/dynamic_1/wlp/bin` directory.

__ c. To stop the server, enter the following command:

```
./server stop dynamic_1
```

__ d. Go to the `/opt/wlp/packagedServers/dynamic_2/wlp/bin` directory.

__ e. To stop the server, enter the following command:

```
./server stop dynamic_2
```

__ f. Go to the `/opt/wlp/packagedServers/dynamic_3/wlp/bin` directory.

__ g. To stop the server, enter the following command:

```
./server stop dynamic_3
```

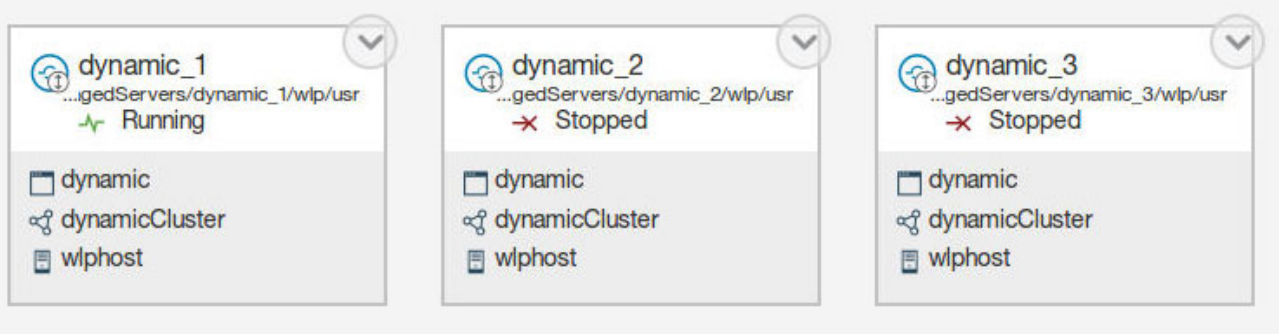
The controller might have already stopped the server due to the defined scaling policy.

__ h. Examine the output in the tail window from the `messages.log` file.

```
[9/15/16 8:47:40:918 EDT] 000007e4 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0115I: The scaling controller is stopping server dyna
3 in user directory /opt/wlp/packagedServers/dynamic_3/wlp/usr on host wlpho
o reduce capacity in cluster dynamicCluster.
[9/15/16 8:47:46:335 EDT] 000007e4 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0114I: The scaling controller has successfully stoppe
rver dynamic_3 on host wlphost.
[9/15/16 8:48:11:922 EDT] 000007f1 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0111I: The scaling controller is starting server dyna
1 in user directory /opt/wlp/packagedServers/dynamic_1/wlp/usr on host wlpho
o meet the minimum instances for cluster dynamicCluster.
[9/15/16 8:48:19:846 EDT] 000007f7 ctive.repository.internal.metadata.AdminM
ataEventHandler I CWWKX9068I: Administrative metadata for resource wlphost,/
wlp/packagedServers/dynamic_1/wlp/usr,dynamic_1,dynamic was removed from the
lective repository.
[9/15/16 8:48:20:164 EDT] 000007f1 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0112I: The scaling controller has successfully starte
rver dynamic_1 on host wlphost.
```

__ i. Maximize the Admin Center.

__ j. On the Servers pane, examine the servers that are running and the servers that are stopped.



__ 2. Change the defined scaling policy for the controller.

__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Go to the scalingDefinitions section.

__ e. Change the minimum instance number for the defaultScalingPolicy to 2. To modify the defaultScalingPolicy, use the following details:

```
<defaultScalingPolicy min="2" max="4" enabled="true">
```

```
<scalingDefinitions>
        <defaultScalingPolicy min="2" max="4" enabled="true">
                <metric name="cpu" min="30" max="80"/>
                <metric name="heap" min="30" max="80"/>
                <metric name="memory" min="30" max="80"/>
                <in units="instance" amount="1" minInterval="30s"/>
                <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

__ f. Save the changes.

__ g. Do not close the file. You continue to update this file through the remainder of the exercise.

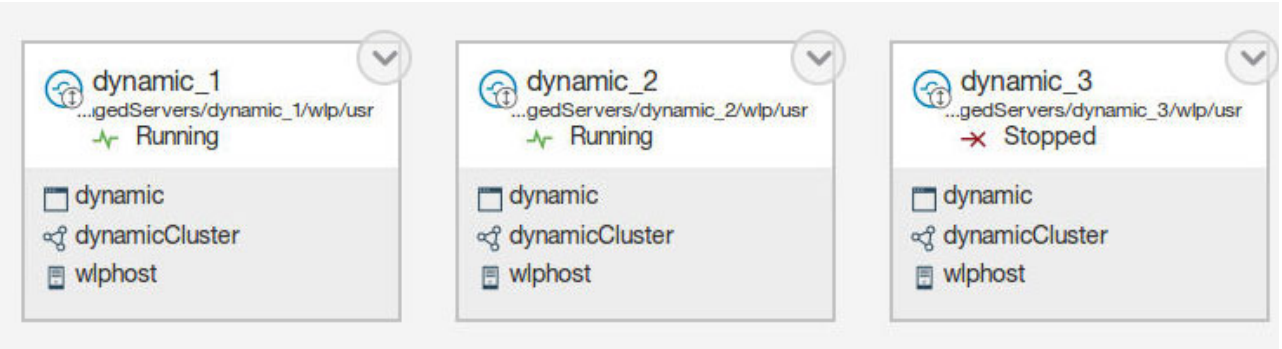
__ 3. Verify that the controller started one more instance.

__ a. Go to the terminal window where the tail of the message.log file is running. You should see that the controller started another instance. The instance that is started might be different from the one noted in the screen capture.

```
[9/15/16 9:02:49:780 EDT] 000008fa com.ibm.ws.config.xml.internal.ConfigRefr
r               A CWWKG0017I: The server configuration was successfully upda
in 0.168 seconds.
[9/15/16 9:03:00:788 EDT] 00000902 ctive.repository.internal.metadata.AdminN
ataEventHandler I CWWKX9068I: Administrative metadata for resource wlphost,/
wlp/packagedServers/dynamic_2/wlp/usr,dynamic_2,dynamic was removed from the
lective repository.
[9/15/16 9:03:03:134 EDT] 000008f9 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0112I: The scaling controller has successfully starte
rver dynamic_2 on host wlphost.
```

__ b. Maximize the Admin Center.

__ c. On the Servers pane, examine the servers that are running and the servers that are stopped. You should see an extra server that is now running.



Part 5: Testing scaling-in

__ 1. Change the defined scaling policy for the controller.

__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Go to the scalingDefinitions section.

__ e. Change the minimum instance on the defaultScalingPolicy back to 1. Also, change the cpu, heap, and memory min setting to 98 and max setting to 99. The setting of 98% is to have the scale-in capability take place. Update the scalingDefinition with the following information:

```
<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="true" >
             <metric name="cpu" min="98" max="99"/>
             <metric name="heap" min="98" max="99" />
             <metric name="memory" min="98" max="99" />
                   <in units="instance" amount="1" minInterval="30s"/>
                   <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

```
<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="true">
                <metric name="cpu" min="98" max="99"/>
                <metric name="heap" min="98" max="99"/>
                <metric name="memory" min="98" max="99"/>
                <in units="instance" amount="1" minInterval="30s"/>
                <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

__ f. Save the changes.

__ g. Do not close the file. You continue to update this file through the remainder of the exercise.

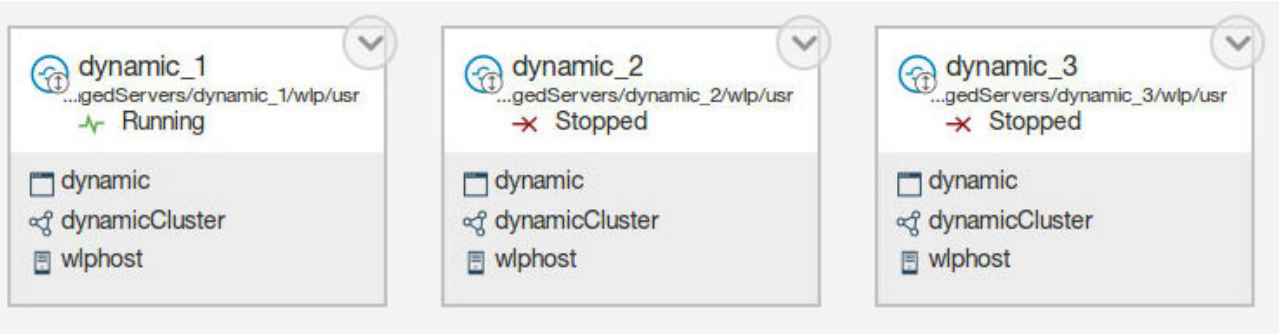
__ 2. Verify that the controller started one more instance.

__ a. Go to the terminal window where the tail of the `messages.log` file is running. You should see that the controller stopped another instance. The instance stopped in your configuration might be different from the one noted in the screen capture.

```
[9/15/16 9:20:31:078 EDT] 00000a33 com.ibm.ws.config.xml.internal.ConfigRefr
r               A CWWKG0017I: The server configuration was successfully upda
in 2.439 seconds.
[9/15/16 9:20:38:076 EDT] 00000a32 com.ibm.ws.scaling.controller.internal.Sc
gExecutorImpl   I CWWKV0114I: The scaling controller has successfully stoppe
rver dynamic_2 on host wlphost.
```

__ b. Maximize the Admin Center.

__ c. On the Servers pane, examine the servers that are running and the servers that are stopped. You should see an extra server that is now stopped.



Part 6: Testing scaling-out

__ 1. Change the defined scaling policy for the controller.

__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. If needed, open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ d. Go to the scalingDefinitions section.

__ e. In a normal environment, increasing user workload increases CPU, total memory, and heap usage to drive a policy breach and cause the controller to start another instance on the same or different hosts.

Due to the constraints of a lab environment, define an artificial policy to cause a policy breach to scale out. This time, make the minimum heap 1%, and maximum heap 2%, which should cause the controller to start more JVM instances. Change the cpu and

memory minimum to 30% and maximum to 80%. Update the scalingDefinition with the following information:

```
<scalingDefinitions>
 <defaultScalingPolicy min="1" max="4" enabled="true" >
        <metric name="cpu" min="30" max="80"/>
        <metric name="heap" min="1" max="2" />
        <metric name="memory" min="30" max="80" />
             <in units="instance" amount="1" minInterval="30s"/>
             <out amount="1" units="instance" minInterval="30s" />
 </defaultScalingPolicy>
</scalingDefinitions>
```

```
<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="true">
                <metric name="cpu" min="30" max="80"/>
                <metric name="heap" min="1" max="2"/>
                <metric name="memory" min="30" max="80"/>
                <in units="instance" amount="1" minInterval="30s"/>
                <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
</scalingDefinitions>
```

__ f. Save the changes.

__ g. Do not close the file. You continue to update this file through the remainder of the exercise.

__ 2. Verify that the controller started one more instance.

__ a. Go to the terminal window where the tail of the `messages.log` file is running. You should see that the controller started another instance. The instance that is started might be different from the one noted in the screen capture.

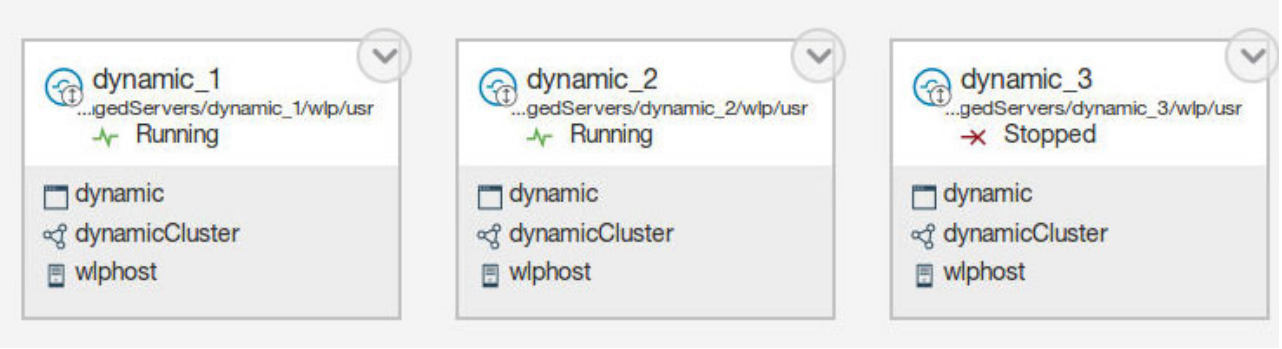
```
5/16 9:28:28:245 EDT] 00000acf com.ibm.ws.config.xml.internal.ConfigRef
            A CWWKG0017I: The server configuration was successfully upd
.556 seconds.
5/16 9:28:36:579 EDT] 00000ad5 com.ibm.ws.scaling.controller.internal.S
cutorImpl   I CWWKV0113I: The scaling controller is starting server dyn
 user directory /opt/wlp/packagedServers/dynamic_2/wlp/usr on host wlph
crease capacity in cluster dynamicCluster.  The average heap utilizatio
cluster is 4.028 percent.
5/16 9:28:48:166 EDT] 00000ae0 ctive.repository.internal.metadata.Admin
ventHandler I CWWKX9068I: Administrative metadata for resource wlphost,
packagedServers/dynamic_2/wlp/usr,dynamic_2,dynamic was removed from th
ive repository.
5/16 9:28:50:494 EDT] 00000ad5 com.ibm.ws.scaling.controller.internal.S
cutorImpl   I CWWKV0112I: The scaling controller has successfully start
 dynamic_2 on host wlphost.
```

__ b. Look for a message that indicates the average heap use for the cluster.

__ c. Observe the output for a few minutes. You can see that the controller starts another instance.

__ d. Maximize the Admin Center.

__ e. On the Servers pane, examine the servers that are running and the servers that are stopped. You should see an extra server that is now running.



__ f. Let the configuration run for a few minutes and the controller starts an extra server.

Part 7: Testing a cold start

__ 1. Log out of the Admin Center.

__ a. Log out of the Admin Center.

__ b. Minimize the browser window.

__ 2. Change the defined scaling policy for the controller.

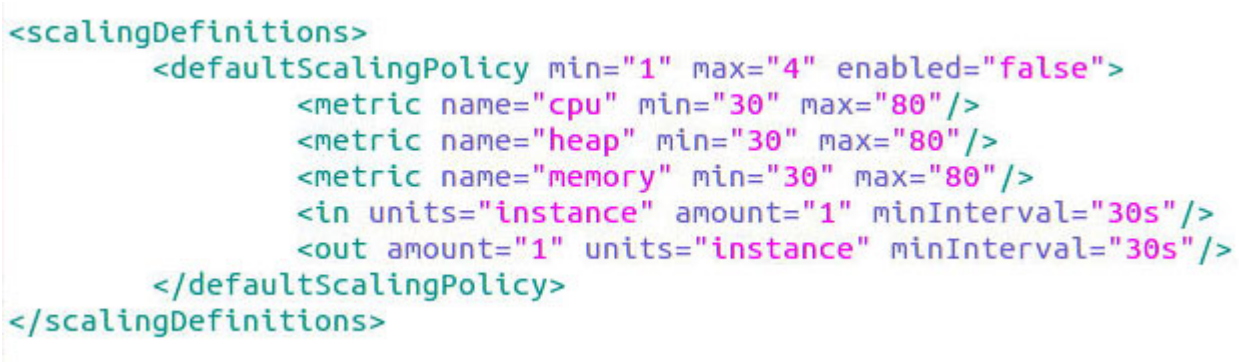
__ a. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ b. If needed, open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ c. Go to the scalingDefinitions section.

__ d. Reset the scaling policy to the initial one that is used for this exercise. Also, disable scaling by changing enabled to false. Update the scalingDefinition with the following information:

```
<scalingDefinitions>
        <defaultScalingPolicy min="1" max="4" enabled="false" >
             <metric name="cpu" min="30" max="80"/>
             <metric name="heap" min="30" max="80" />
             <metric name="memory" min="30" max="80" />
                    <in units="instance" amount="1" minInterval="30s"/>
                    <out amount="1" units="instance" minInterval="30s"/>
        </defaultScalingPolicy>
 </scalingDefinitions>
```



Important

Make sure that you disable scaling by using `enabled="false"`. Otherwise, when you stop servers in the cluster, the controller starts a member per the defined scaling policy.

__ e. Save the changes.

__ f. Do not close the file. You continue to update this file through the remainder of the exercise.

__ 3. Stop all the servers in the dynamic cluster by using the command line.

__ a. Go to the `/opt/wlp/packagedServers/dynamic_1/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop dynamic_1
```

__ c. Go to the `/opt/wlp/packagedServers/dynamic_2/wlp/bin` directory.

__ d. To stop the server, enter the following command:

```
./server stop dynamic_2
```

__ e. Go to the `/opt/wlp/packagedServers/dynamic_3/wlp/bin` directory.

__ f. To stop the server, enter the following command:

```
./server stop dynamic_3
```

__ 4. Stop the adminCenterController by using the command line.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop adminCenterController
```

__ 5. Enable the scaling policy.

__ a. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ b. If needed, open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ c. Go to the scalingDefinitions section.

__ d. Enable scaling by changing enabled to true. Update the scalingDefinition with the following information:

```
<defaultScalingPolicy min="1" max="4" enabled="true" >
```

__ e. Save the changes.

__ f. Do not close the file. You continue to update this file through the remainder of the exercise.

__ 6. Start the adminCenterController and log in to the Admin Center.

__ a. To start the server, enter the following command:

```
./server start adminCenterController --clean
```

__ b. Maximize the Admin Center browser.

__ c. In the login area, enter `admin` as the User Name and `passw0rd` as the Password and click **Submit**.

__ 7. Examine the configuration.

__ a. Click **Explore**.

__ b. Click the **SERVERS** pane.

__ c. Wait a few moments and verify that the controller starts at least one member.

__ d. Minimize the Admin Center.

__ e. Go to the terminal window where the tail of the `messages.log` file is running. You should see that the controller started another instance. If needed, restart the tail process to see the latest information.

__ 8. Change the defined scaling policy for the controller.

__ a. If needed, open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ b. Go to the scalingDefinitions section.

__ c. Next, use a cluster-specific policy by using the bind clusters option. Also, change the minimum value 2. Update the scalingDefinition with the following information:

```
<scalingPolicy min="2" max="4" enabled="true">
        <bind clusters="dynamicCluster"/>
```

__ d. Save the changes.

__ e. Do not close the file. You continue to update this file through the remainder of the exercise.

__ f. Go to the terminal window where the tail of the `messages.log` file is running. You should see that the controller started another instance.

__ g. Maximize the Admin Center.

__ h. Verify that another server is started.

__ i. When completed, log out of the Admin Center.

__ j. Close the browser window.

Part 8: Cleaning up the environment

In this part of the exercise, you clean up the deployed artifacts.

__ 1. Change the defined scaling policy for the controller.

__ a. Go to a terminal window.

__ b. Go to the `/opt/wlp/usr/servers/adminCenterController` directory.

__ c. If needed, open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ d. Go to the scalingDefinitions section.

__ e. Disable scaling by changing enabled to false. Update the scalingDefinition with the following information:

```
<defaultScalingPolicy min="1" max="4" enabled="false" >
```

__ f. Save and close the file.

__ 2. Stop the adminCenterController by using the command line.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop adminCenterController
```

__ 3. Stop all the servers in the dynamic cluster by using the command line.

__ a. Go to the `/opt/wlp/packagedServers/dynamic_1/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop dynamic_1
```

__ c. Go to the `/opt/wlp/packagedServers/dynamic_2/wlp/bin` directory.

__ d. To stop the server, enter the following command:

```
./server stop dynamic_2
```

__ e. Go to the `/opt/wlp/packagedServers/dynamic_3/wlp/bin` directory.

__ f. To stop the server, enter the following command:

```
./server stop dynamic_3
```

__ g. Exit the terminal window.

__ h. Exit the tail command in the terminal window. Exit the terminal window.

End of exercise

Exercise review and wrap-up

In this exercise, you enabled and tested the autonomic scaling capability of Liberty servers.

Exercise 5. Using the IBM HTTP Server with SSL to a Liberty server

Estimated time

01:15

Overview

In this exercise, you configure Liberty, the IBM HTTP Server, and the plug-in that connects them to use SSL. You secure communication from the browser to the web server, the plug-in, and onto Liberty.

Objectives

After completing this exercise, you should be able to:

- Deploy a simple application that displays protocol information
- Configure Liberty to use SSL
- Configure the IBM HTTP Server to use SSL
- Configure the plug-in between the IBM HTTP Server and Liberty
- Configure SSL between the plug-in and Liberty

Introduction

Communications are secured with Secure Sockets Layer (SSL) protocol. The SSL protocol provides transport layer security, which includes authenticity, data signing, and data encryption to ensure a secure connection between a client and server that uses WebSphere Application Server. The foundation technology for SSL is public key cryptography, which guarantees that when an entity encrypts data by using its public key, only entities with the corresponding private key can decrypt that data. The Liberty Server uses Java Secure Sockets Extension (JSSE) as the SSL implementation for secure connections. JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure that secure connectivity exists across most protocols. JSSE relies on X.509 certificate-based asymmetric key pairs for secure connection protection and some data encryption. Key pairs effectively encrypt session-based secret keys that encrypt larger blocks of data. The SSL implementation manages the X.509 certificates.

Requirements

To complete this exercise, you need the WebSphere Liberty binary files and Liberty installed. You also need the IBM HTTP Server binary files and the web server installed.

Exercise instructions

In this exercise, you secure the communication in the environment. The following steps are completed:

- Configure and test SSL for Liberty
- Configure the IBM HTTP Server for SSL
- Configure the plug-in for the IBM HTTP Server
- Generate a `plugin-cfg.xml` file
- Configure SSL between the plug-in and Liberty

Part 1: Creating the SSLServer

__ 1. Create the server, SSLServer.

__ a. Open a terminal window.

__ b. Go to the `/opt/wlp/bin` directory.

__ c. To create the server, SSLServer, enter the following command:

```
./server create SSLServer
```

__ d. Go to the `/opt/wlp/usr/servers/SSLServer` directory.

__ e. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ f. Go to the httpEndpoint section. Notice that the httpPort is 9080 and httpsPort is 9443.

server.xml ×

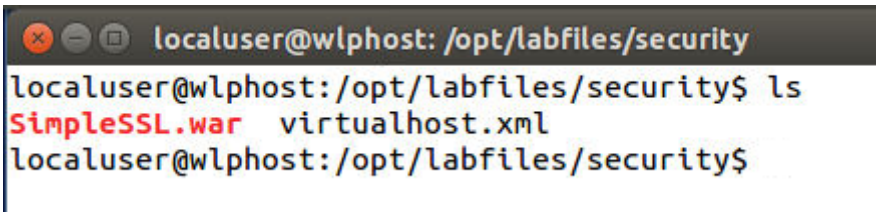
```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host
attribute to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />
```

__ g. Close the file when completed.

__ 2. Deploy the SimpleSSL sample.

__ a. Go to the `/opt/labfiles/security` directory.

__ b. List the contents of the directory. There is a `SimpleSSL.war` file.



__ c. Copy the file to the SSLServer dropins directory. To copy, enter the following command:

```
cp SimpleSSL.war /opt/wlp/usr/servers/SSLServer/dropins/
```

__ 3. Start the SSLServer.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To start the server, enter the following command:

```
./server start SSLServer
```

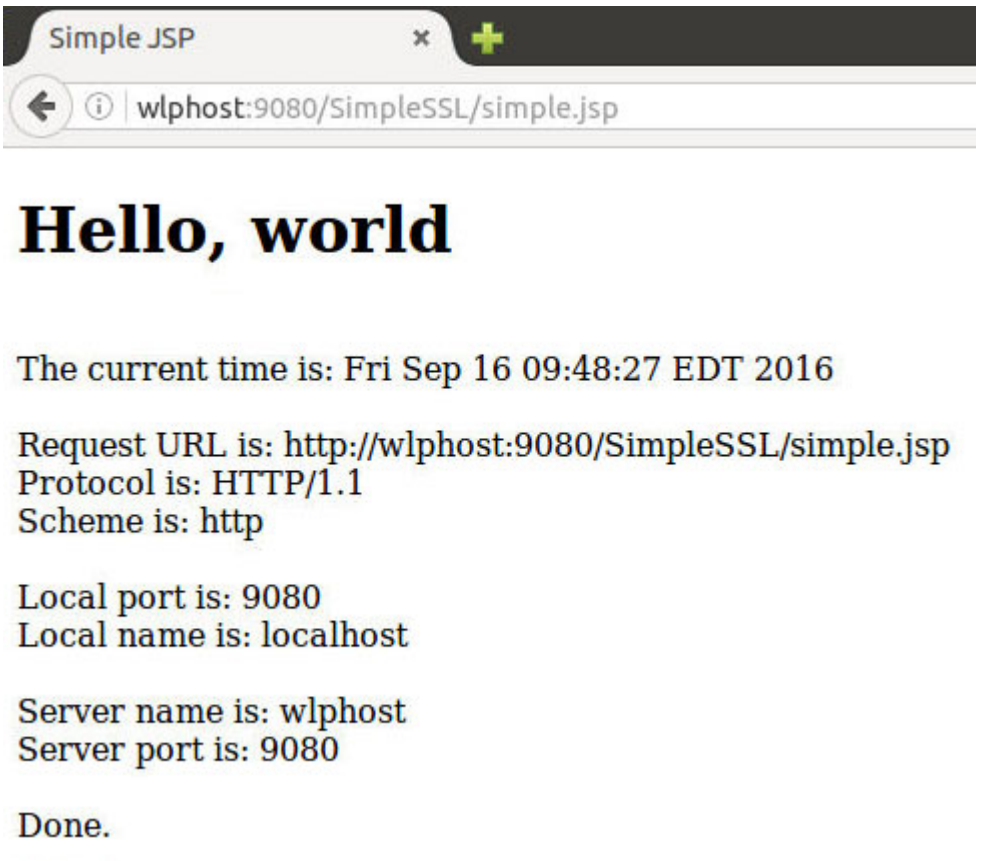
Wait for the server to start.

__ 4. Test the configuration.

__ a. Open a Firefox browser.

__ b. Test the application by entering the following URL:

```
http://wlphost:9080/SimpleSSL/simple.jsp
```



You can see that the sample application prints the URL scheme, which is HTTP. The local port for the application is 9080, and the server port in the URL, which is the same 9080.

__ c. Minimize the Firefox browser.

Part 2: Configuring Liberty to use HTTPS/SSL

__ 1. Edit the configuration file for the SSLServer.

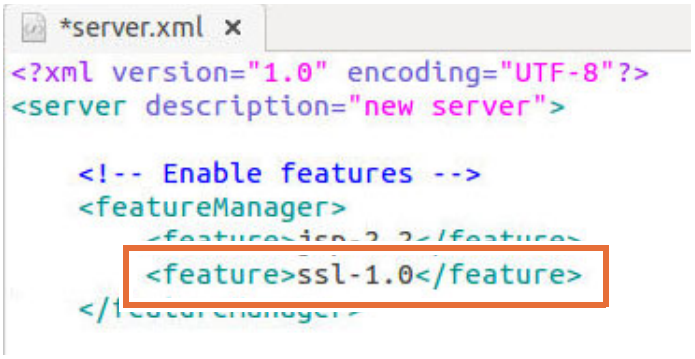
__ a. Go to the terminal window.

__ b. Go to the `/opt/wlp/usr/servers/SSLServer` directory.

__ c. Open the `server.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

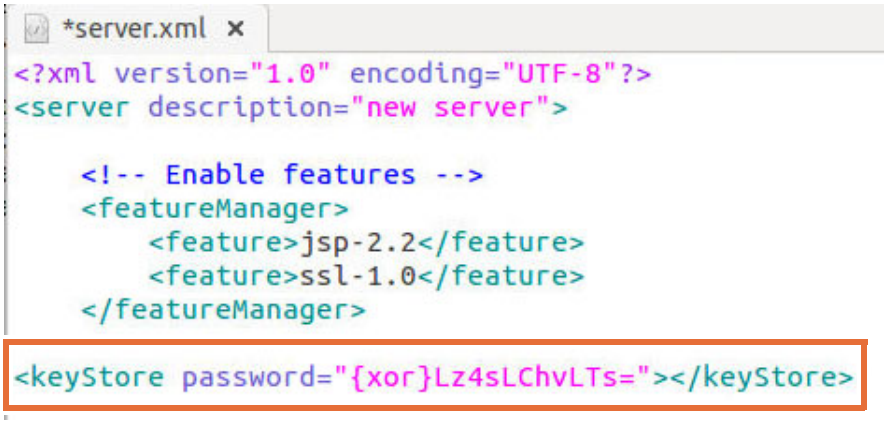
__ d. Go to the feature manager section. Add the following feature:

```
<feature>ssl-1.0</feature>
```



__ e. Add a keystore definition and use the encoded version of the password, `passw0rd`. Add the definition after the featureManager section. To enter the keystore definition, enter the following information:

```
<keyStore password="{xor}Lz4sLChvLTs="></keyStore>
```

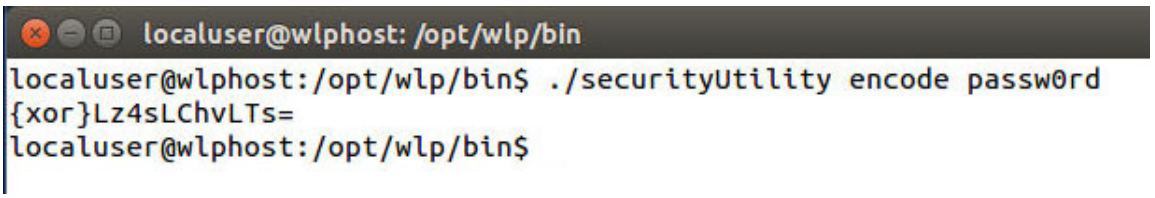


This is the encoded password that you generated in Exercise 1.

Reminder

You can encode a password by using the `securityUtility` command. To encode the admin password, `passw0rd`, enter the following command:

```
./securityUtility encode passw0rd
```



Note the encoded password that is displayed. This encoded password can be placed in various configuration files that you use in a Liberty environment.

__ f. Save the file.

__ g. When completed, close the file.

__ 2. Examine the log file for the SSLServer.

__ a. In the terminal window, go to the `/opt/wlp/usr/servers/SSLServer/logs` directory.

__ b. Open the `messages.log` file. Open the file by using an editor such as vi or gedit.

__ c. Go to the bottom of the file. Look for a message that the server installed the `ssl-1.0` feature.

__ d. When completed, close the file.

__ 3. Test the configuration.

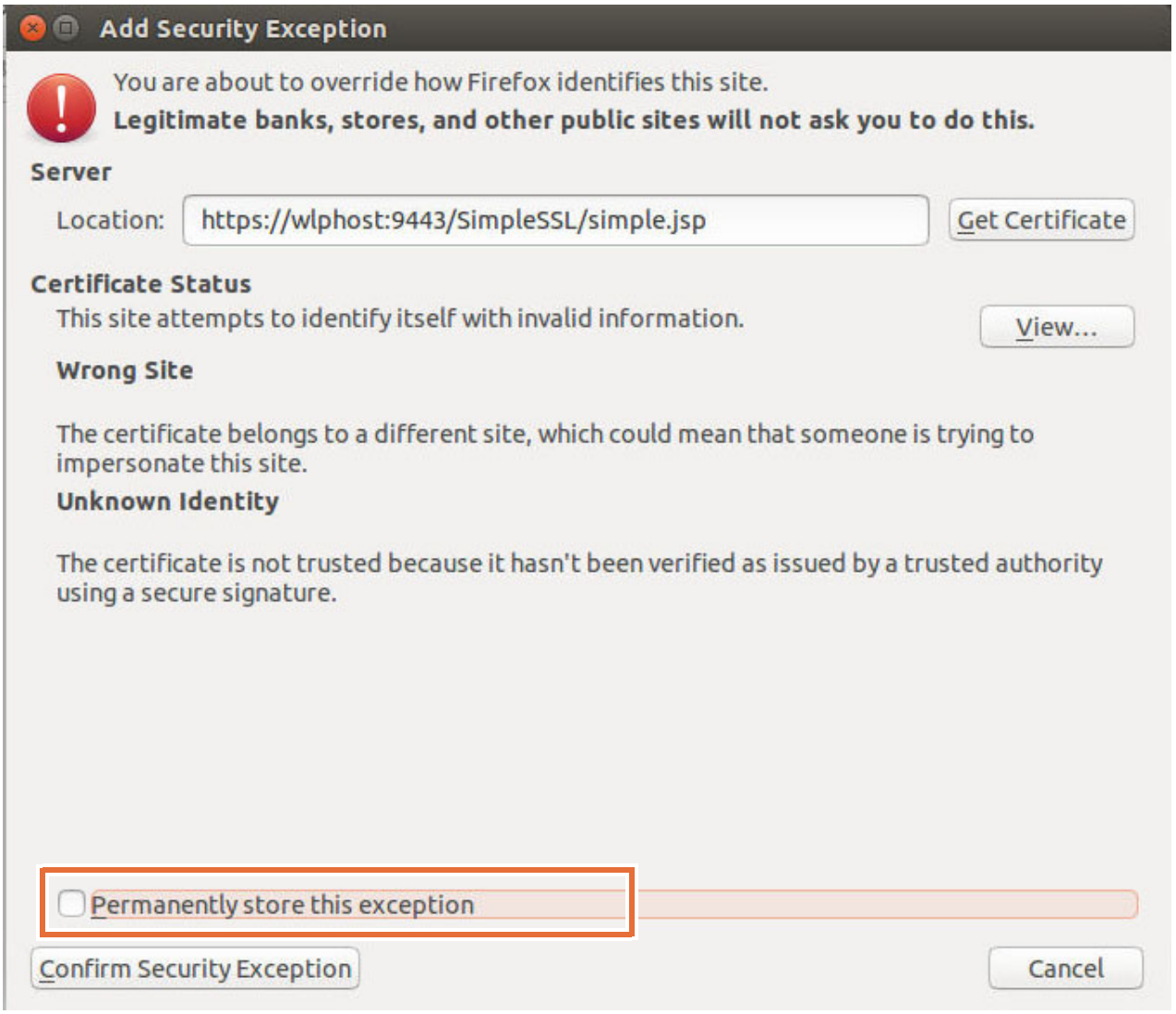
__ a. Maximize the browser window.

__ b. Test the application. This time, use the HTTPS port for the server. To test the application, enter the following URL:

`https://wlphost:9443/SimpleSSL/simple.jsp`

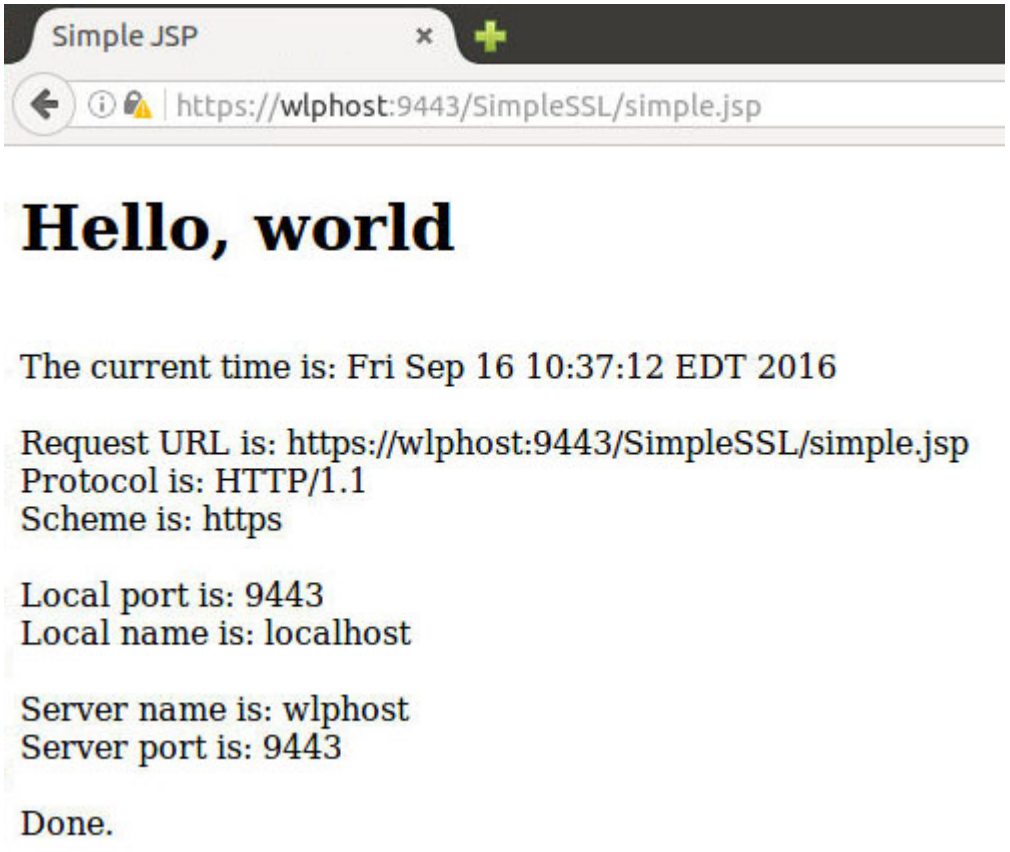
__ c. In the Your connection is not secure pane, click **Advanced**. Click **Add Exception**.

__ d. Clear the **Permanently store this exception** check box.



__ e. Click **Confirm Security Exception**.

__ f. The page is loaded and the scheme is now https. The local port for the application is 9443, and the server port in the URL, which is the same 9443.



This output shows that the Liberty server can run a simple JSP by using either HTTP (non-secure) by using port 9080, or using HTTPS / SSL (secure) by using port 9443.

__ g. Minimize the browser window.

Part 3: Configuring the IBM HTTP Server to use HTTPS/SSL

The IBM HTTP Server is already installed on the course image. By default, the IBM HTTP Server listens on port 80. In an earlier exercise, the port was changed to 9180. In this part of the exercise, you examine the settings for the IBM HTTP Server and test that it is running.

__ 1. Verify the details in the `httpd.conf` file.

__ a. Go to the terminal window.

__ b. Go to `/opt/IBM/HTTPServer/conf` directory.

__ c. Open the `httpd.conf` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Verify that Listen is set to `9180`. This port was set in an earlier exercise.

```
# Listen: Allows you to bind the web server to specific IP addresses
# and/or ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent the web server from accepting connections on all interfaces
# (0.0.0.0)
#
# Change this to "Listen 0.0.0.0:port" to restrict the server to
# IPv4.
#
#Listen 12.34.56.78:80
Listen 9180
```

__ e. When completed, close the file.

__ 2. Start the IBM HTTP Server.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. To start the HTTP Server, enter the following command:

```
./apachectl start
```

__ c. To see whether the HTTP Server is started, enter the following command:

```
ps -ef | grep httpd
```

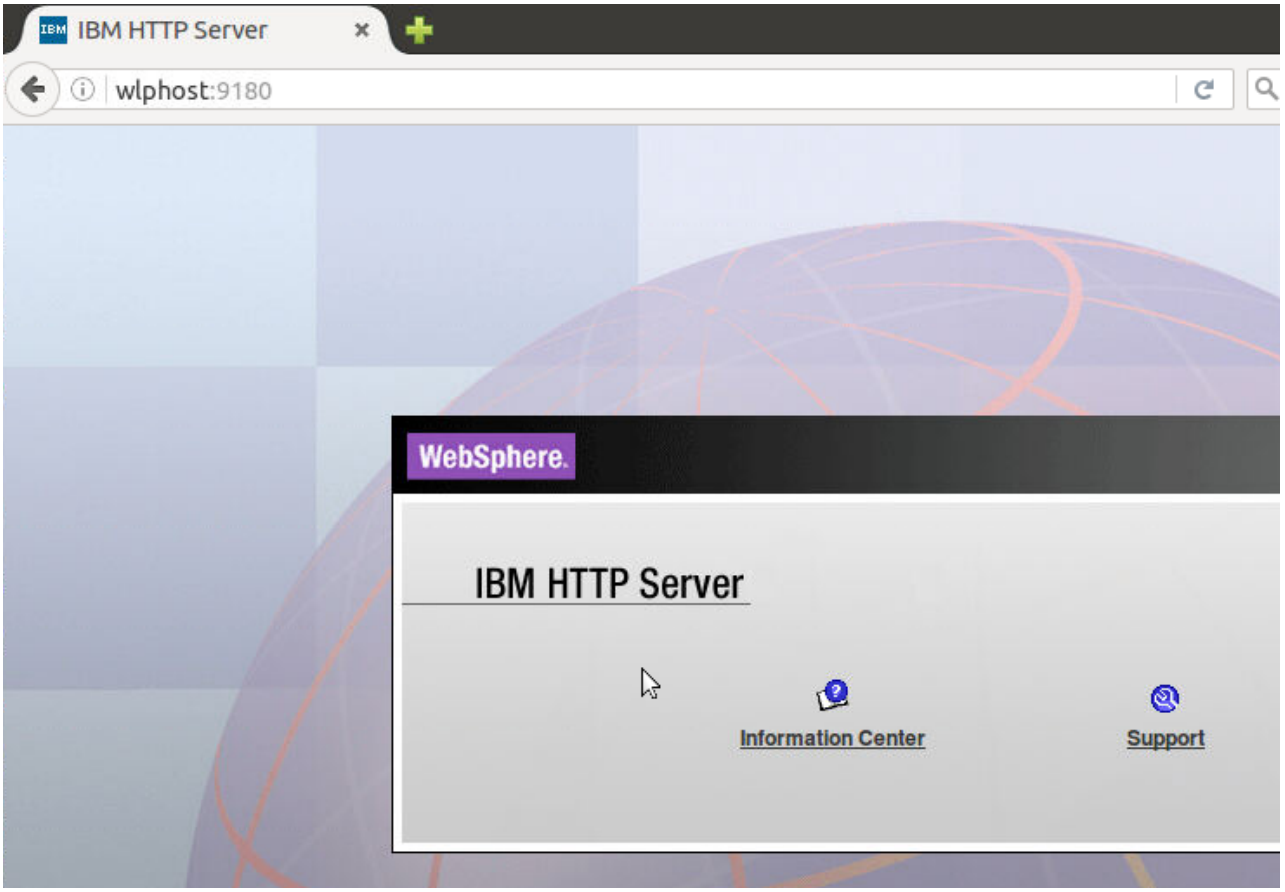
```
localuser@wlphost: /opt/IBM/HTTPServer/bin
ocaluser@wlphost:/opt/IBM/HTTPServer/bin$ ps -ef | grep httpd
ocalus+  8816  1672  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd
/opt/IBM/HTTPServer -k start
ocalus+  8818  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd
/opt/IBM/HTTPServer -k start
ocalus+  8819  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd
/opt/IBM/HTTPServer -k start
ocalus+  8820  8816  0 09:08 ?        00:00:00 /opt/IBM/HTTPServer/bin/httpd
/opt/IBM/HTTPServer -k start
ocalus+  8927 18692  0 09:08 pts/26   00:00:00 grep --color=auto httpd
ocalus+ 19975 18692  0 Sep02 pts/26   00:00:09 gedit httpd.conf
ocaluser@wlphost:/opt/IBM/HTTPServer/bin$
```

You can see that a number of processes are started.

__ d. Maximize the browser window.

__ e. Test the web server by entering the following URL:

```
http://wlphost:9180
```



You can see the main page for the IBM HTTP Server.

__ f. Close the browser window.

__ 3. Stop the IBM HTTP Server.

__ a. Go to the terminal window.

__ b. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ 4. Configure the IBM HTTP Server to use SSL. Create a key ring with a self-signed certificate for IBM HTTP Server.

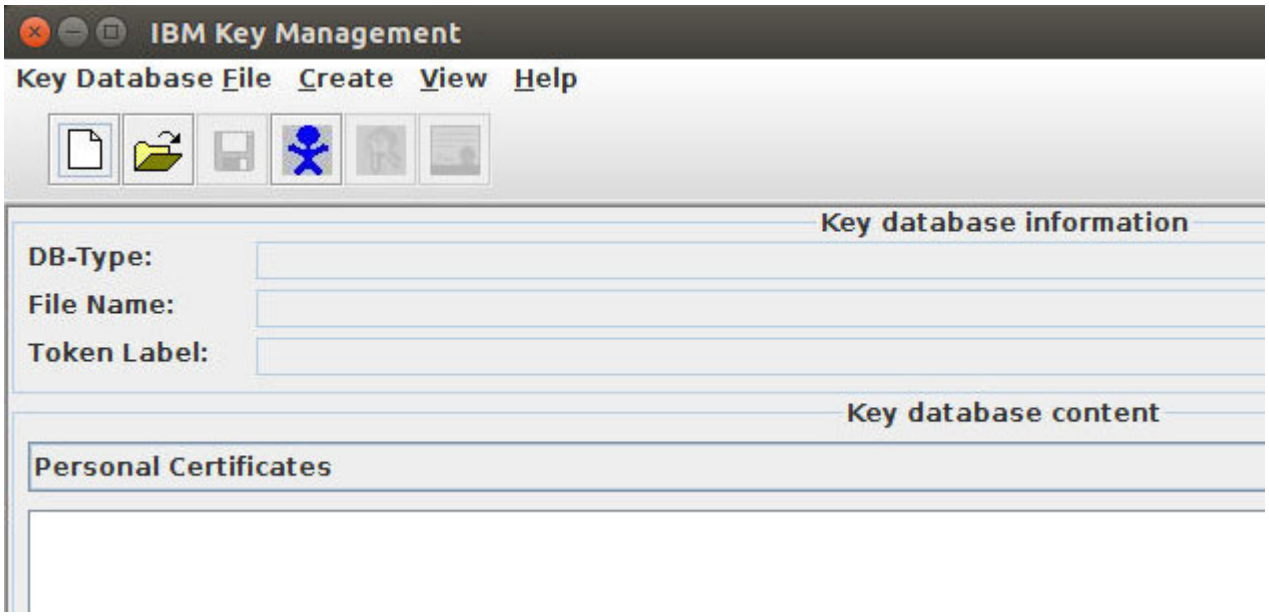
__ a. Go to the `/opt/IBM/HTTPServer` directory.

__ b. Create the directory `ssl`. The complete path would be `/opt/IBM/HTTPServer/ssl`.

__ c. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ d. Start iKeyman by entering the following command:

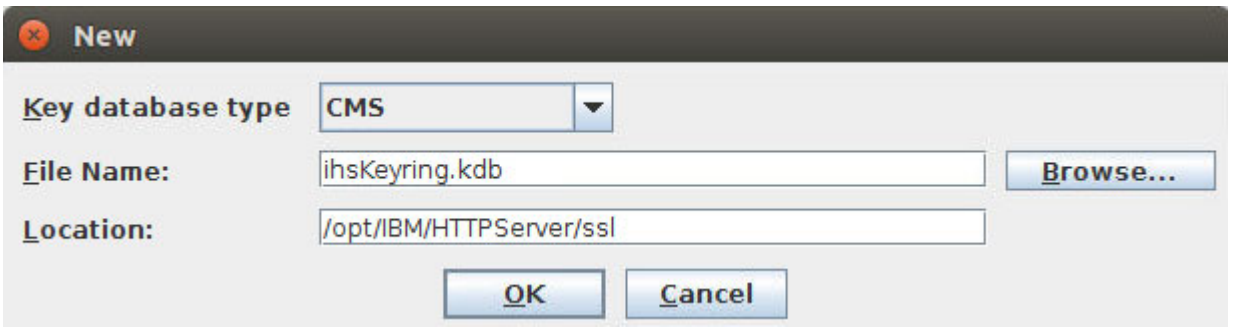
```
./ikeyman
```



__ e. Create a new key ring by clicking **Key Database File > New**.

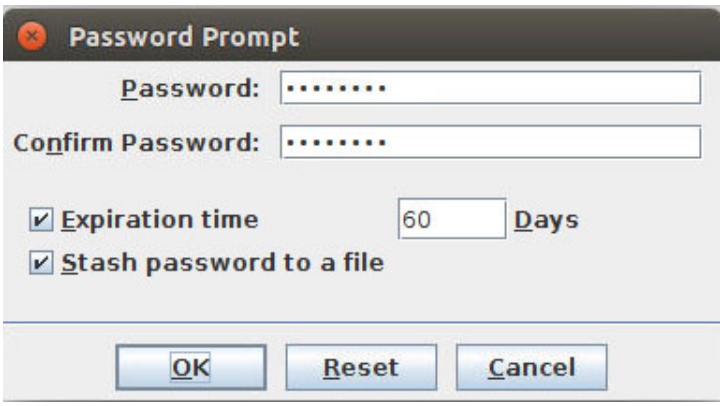
__ f. Enter the following information:

- In the **Key database type** field, select `CMS`
- In the **File Name** field, enter: `ihsKeyring.kdb`
- In the **Location** field, enter: `/opt/IBM/HTTPServer/ssl`



__ g. Click **OK**.

__ h. When prompted for a password for the key ring, enter and confirm `passw0rd` as the password. If wanted, select the **Expiration time** check box and the **Stash password to a file** check box.



Information

When the stash file is created, it contains an encoded form of the password. This encoding prevents casual viewing of the password, but is not highly secure. Therefore, you should protect this file by using operating system file permissions to prevent all access from unauthorized principals.

The file name of the stash file is the same as the name of the key file, only it has a `.sth` suffix. The stash file gets stored in the same directory as the key file.

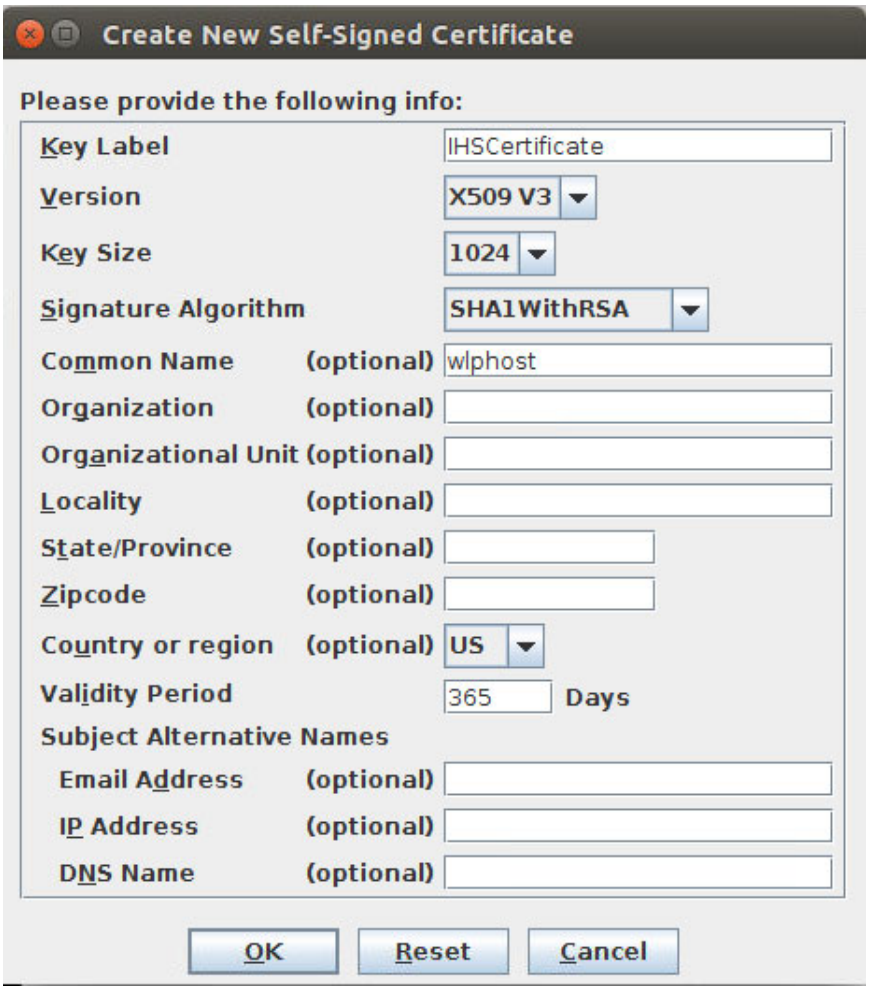
__ i. Click **OK**.

__ 5. Create a self-signed certificate.

__ a. In iKeyman, click **Create > New Self-Signed Certificate**.

__ b. Enter the following information:

- In the **Key Label** field, enter: `IHSCertificate`
- In the **Common Name** field, enter: `wlphost`



__ c. Accept the defaults for the Version, Key Size, and Validity Period.

__ d. Click **OK**.

Information

This process stores the certificate in the key file in the Personal Certificates section. Optionally, it is possible to extract the public signing certificate so that clients can use it. To extract, click **Extract Certificate**, and then enter a **File Name** and **Location**. Click **OK**.

__ e. Exit iKeyman by clicking **Key Database File > Exit**.

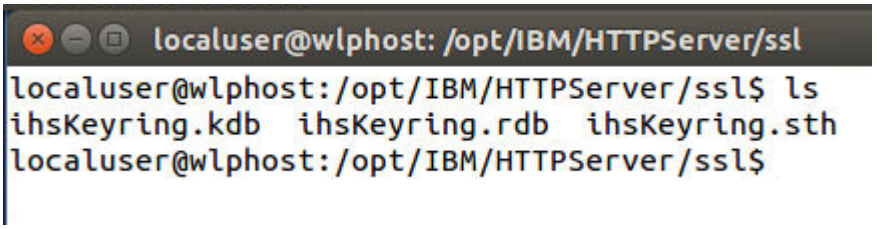
__ 6. Examine the configuration.

__ a. Go to a terminal window.

__ b. Go to the `/opt/IBM/HTTPServer/ssl` directory.

__ c. List the contents of the directory and verify that the following files are created:

- `ihsKeyring.kdb`
- `ihsKeyring.sth`
- `ihsKeyring.rdb`



__ 7. Configure the IBM HTTP Server for HTTPS. Configuration requires that you modify the `httpd.conf` to define the required setting to enable SSL for IBM HTTP Server. It also includes loading the SSL module, defining a listener port, defining a virtual host, and enabling SSL.

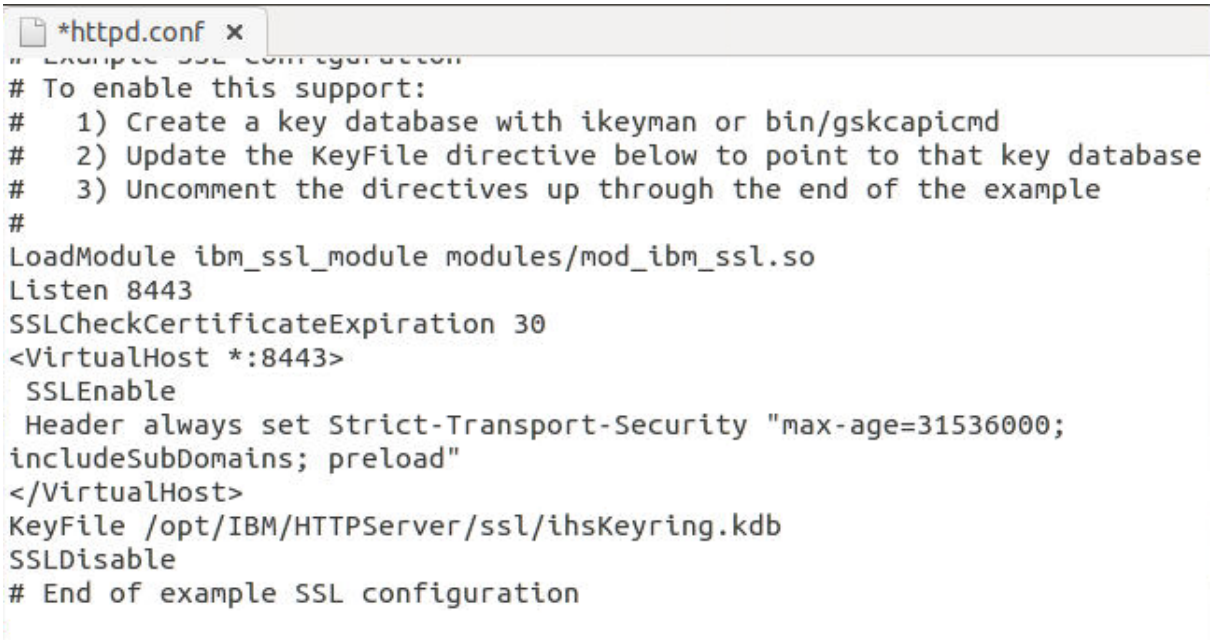
__ a. Go to the `/opt/IBM/HTTPServer/conf` directory.

__ b. Open the `httpd.conf` file. Open the file by using an editor such as vi or gedit.

__ c. Go to the SSL configuration section. You can search for **SSLEnable** to quickly locate the section.

__ d. Uncomment the directives in the SSL configuration area. Then, edit the port to 8443 and edit the KeyFile line so they look like the following entries:

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 8443
SSLCheckCertificateExpiration 30
<VirtualHost *:8443>
   SSLEnable
   Header always set Strict-Transport-Security
“max-age=31536000;includeSubDomains; preload”
</VirtualHost>
KeyFile /opt/IBM/HTTPServer/ssl/ihsKeyring.kdb
SSLDisable
```



__ e. Save the changes.

__ f. When completed, close the file.

__ 8. Start the IBM HTTP Server process.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. Enter the following command to start the IBM HTTP Server:

```
./apachectl start
```

__ c. Verify that the IBM HTTP Server process is running by checking the system process list. Enter the following command to verify:

```
ps -ef | grep httpd
```

Hint

If IBM HTTP Server failed to start, check the `/opt/IBM/HTTPServer/logs/error_log` and `/opt/IBM/BPM/profiles/ihsnode01/logs/IHS/httpd_plugin.log` files for the possible cause.

__ 9. Test the configuration. Connect to IBM HTTP Server with HTTPS.

__ a. First, verify that the web server is running. Open a Firefox browser window and enter the following URL:

`http://wlphost:9180`

__ b. Now that the web server is known to be running, verify that HTTPS is working. In the browser, enter the following URL:

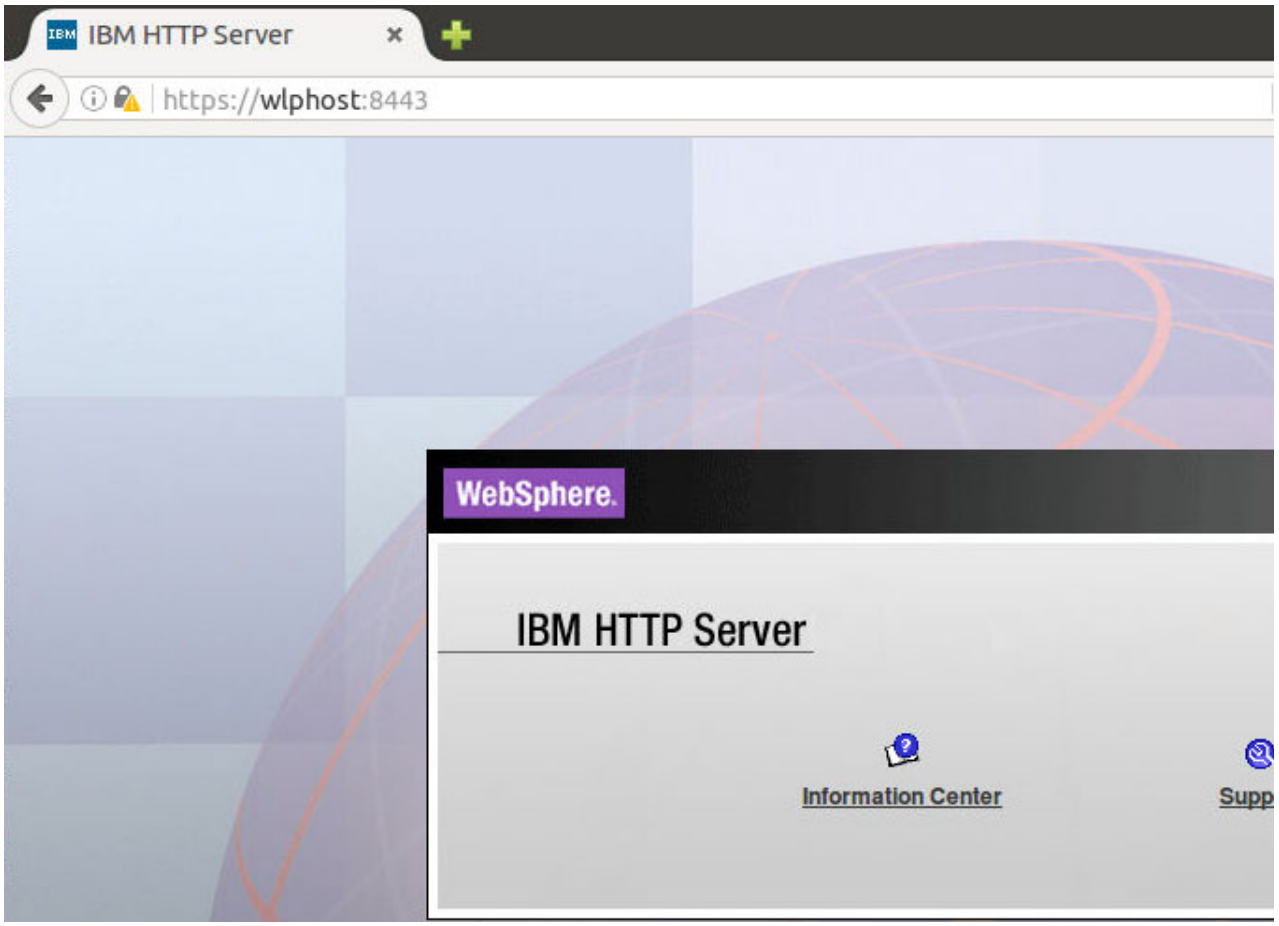
`https://wlphost:8443`

__ c. In the Your connection is not secure pane, click **Advanced**. Click **Add Exception**.

__ d. Clear the **Permanently store this exception** check box.

__ e. Click **Confirm Security Exception**.

__ f. You can see the welcome page for the web server.



__ g. Close the browser window.

Part 4: Configuring the plug-in between IBM HTTP Server and Liberty

You now have a running Liberty server that can be accessed by using either HTTP or HTTPS, and a separate IBM HTTP Server that is also accessible by using either HTTP or by using HTTPS. The two servers have no awareness of each other. The next step is to connect the two, by configuring the web server plug-in so the web server can forward requests for Liberty applications. This configuration allows the web server to be used as a front end for the Liberty server, which can be beneficial for various reasons, including load balancing and high availability in clustered environments.

__ 1. Examine the `httpd.conf` file.

__ a. Go to a terminal window.

__ b. Go to the `/opt/IBM/HTTPServer/conf` directory.

__ c. Open the `httpd.conf` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Go to the bottom of the file.

__ e. Verify that the following two lines are in the file:

```
LoadModule was_ap24_module
/opt/IBM/WebSphere/Plugins/bin/64bits/mod_was_ap24_http.so
WebSpherePluginConfig
/opt/IBM/WebSphere/Plugins/config/IHS1/plugin-cfg.xml
```

```
#LoadModule was_ap24_module /opt/IBM/WebSphere/Plugins/bin/64bits/
mod_was_ap24_http.so
#WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/IHS1/plugin-
cfg.xml

LoadModule was_ap24_module /opt/IBM/WebSphere/Plugins/bin/64bits/
mod_was_ap24_http.so
WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/IHS1/plugin-
cfg.xml
```

The plug-in was configured in a previous lab and this information was added at that time.

Information

These last two lines ultimately cause the web server to forward requests for Liberty applications to the Liberty server. However, you first must generate the plug-in configuration file (`plugin-cfg.xml`) which tells the plug-in what applications belong to which Liberty servers. You generate this file by connecting to Liberty by using the jConsole program. Then, you invoke a particular MBean, which causes Liberty to generate the plug-in configuration file.

__ f. When completed, close the file.

__ 2. Modify the SSLServer configuration file.

__ a. Go to the `/opt/wlp/usr/servers/SSLServer` directory.

__ b. Open the `server.xml` file. Open the file by using an editor such as vi or gedit.

__ c. Go to the end of the httpEndpoint section.

__ d. Open another terminal window and go to the `/opt/labfiles/security` directory.

__ e. Open the `virtualhost.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

virtualhost.xml

```
<!-- Use virtual host default_host for defaultHttpEndpoint -->
    <virtualHost id="default_host"
allowFromEndpointRef="defaultHttpEndpoing">
        <hostAlias>*:9080</hostAlias>
        <hostAlias>*:9443</hostAlias>
        <hostAlias>*:9180</hostAlias>
        <hostAlias>*:8443</hostAlias>
    </virtualHost>
```

__ f. Copy the entire contents of the file.

__ g. Go to the `server.xml` file that is still open.

__ h. Go to the end of the httpEndpoint section.

__ i. Paste the virtual host information into the `server.xml` file. Here, you add the `<virtualHost>` definition, which enables the browser requests coming from any host to be directed towards the ports of 9080 and 9443 for the server, or the ports of 9180 or 8443 for the web server.

*server.xml ×

```
        <feature>jsp-2.2</feature>
        <feature>ssl-1.0</feature>
    </featureManager>

<keyStore password="{xor}Lz4sLChvLTs="></keyStore>

    <!-- To access this server from a remote client add a host
attribute to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />

<!-- Use virtual host default_host for defaultHttpEndpoint -->
    <virtualHost id="default_host"
allowFromEndpointRef="defaultHttpEndpoing">
        <hostAlias>*:9080</hostAlias>
        <hostAlias>*:9443</hostAlias>
        <hostAlias>*:9180</hostAlias>
        <hostAlias>*:8443</hostAlias>
    </virtualHost>
```

Hint

You can also manually enter the following information into the file:

```
<!-- Use virtual host default_host for defaultHttpEndpoint -->
<virtualHost id="default_host" allowFromEndpointRef="defaultHttpEndpoint">
    <hostAlias>*:9080</hostAlias>
    <hostAlias>*:9443</hostAlias>
    <hostAlias>*:9180</hostAlias>
    <hostAlias>*:8443</hostAlias>
</virtualHost>
```

__ j. Save and close the `server.xml` file.

__ k. Close the `virtualhost.xml` file.

__ 3. Restart the server, SSLServer.

__ a. Go to the `/opt/wlp/bin` directory.

__ b. To stop the server, enter the following command:

```
./server stop SSLServer
```

__ c. To start the server, enter the following command:

```
./server start SSLServer
```

Wait for the server to start.

__ 4. Clean up the plug-in directory.

__ a. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ b. List the contents of the directory. There are various files for the plug-in, which includes key files. Remove this content that was created in an earlier exercise. To remove, enter the following command:

```
rm plugin*
```

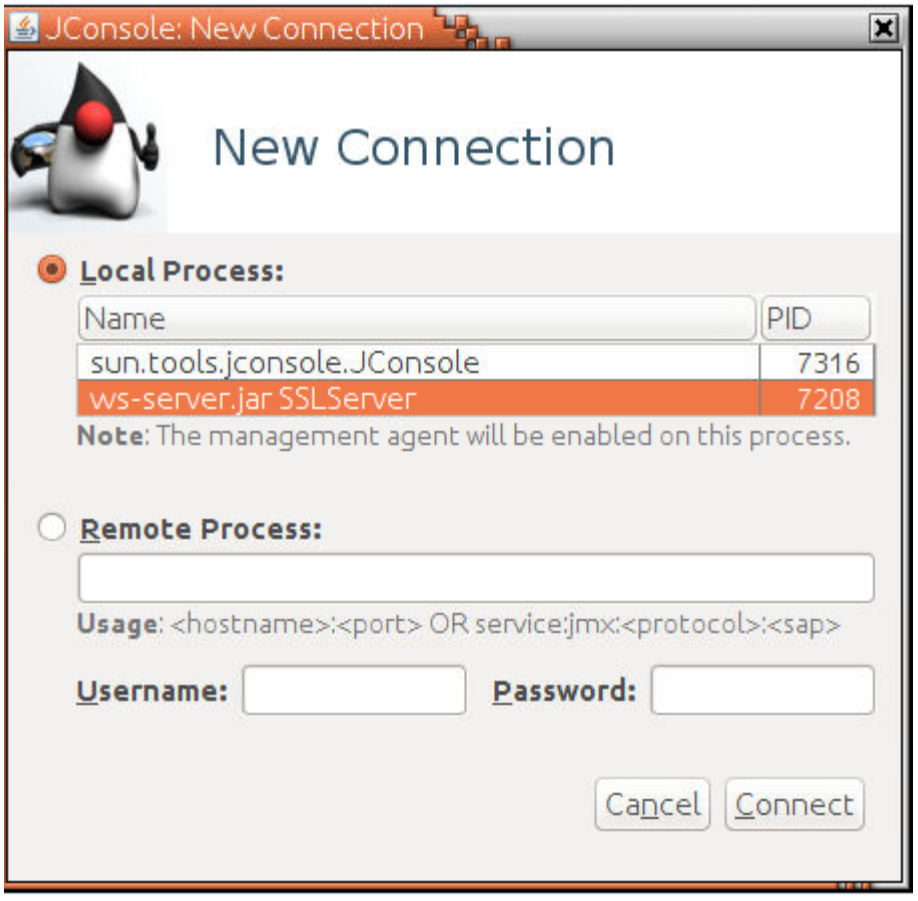
__ 5. Generate the `plugin-cfg.xml` file by using jConsole.

__ a. Go to the `/opt/wlp/java/bin` directory.

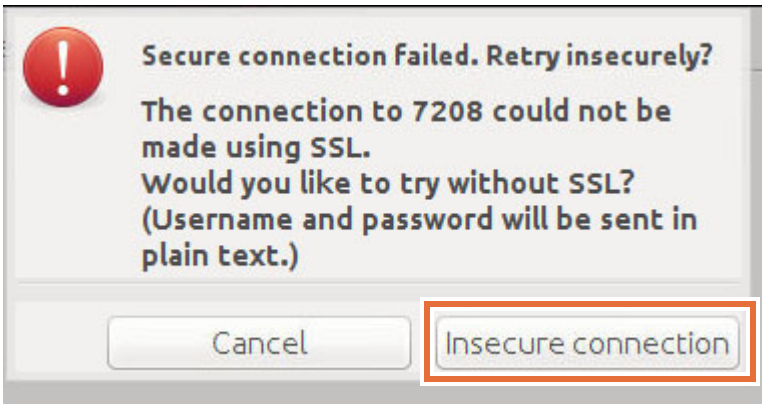
__ b. To start jConsole, enter the following command:

```
./jconsole
```

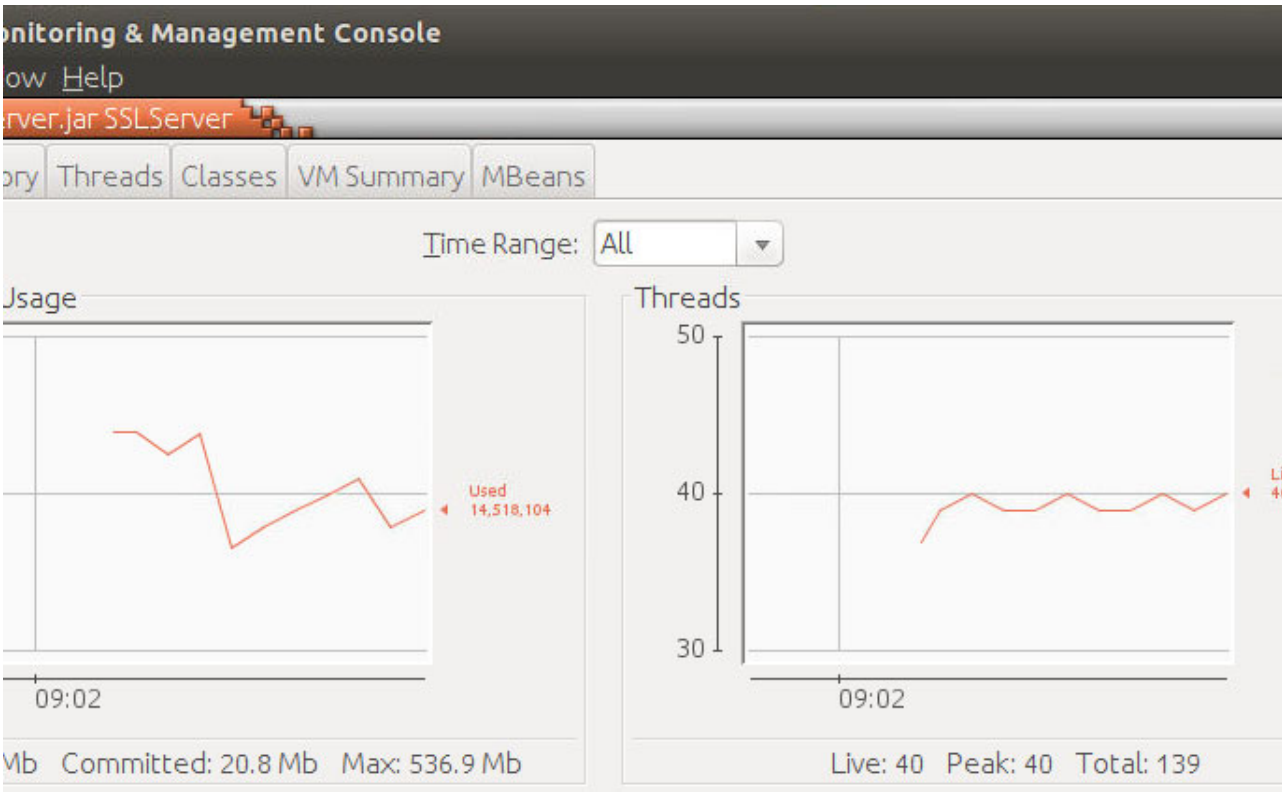
__ c. In the New Connection pane, go to the Local Process section. Select **ws-server.jar SSLServer** and click **Connect**.



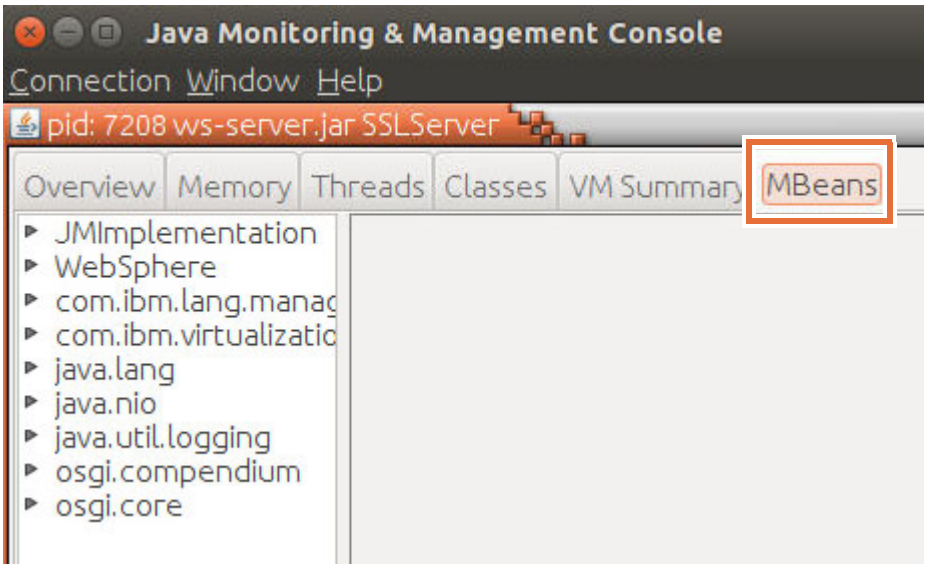
__ d. In the window that appears, click **Insecure connection**.



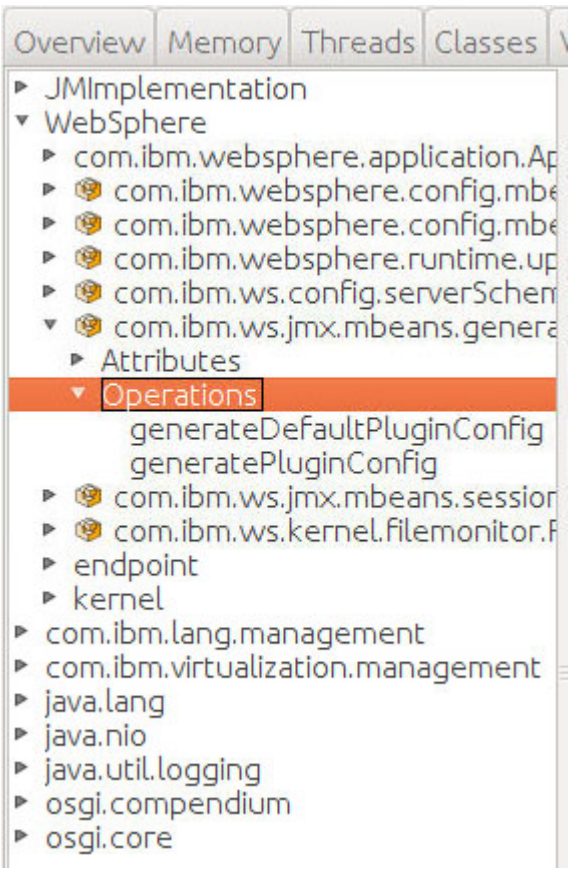
__ e. The jConsole Monitoring and Management Console appears and looks like the following screen capture.



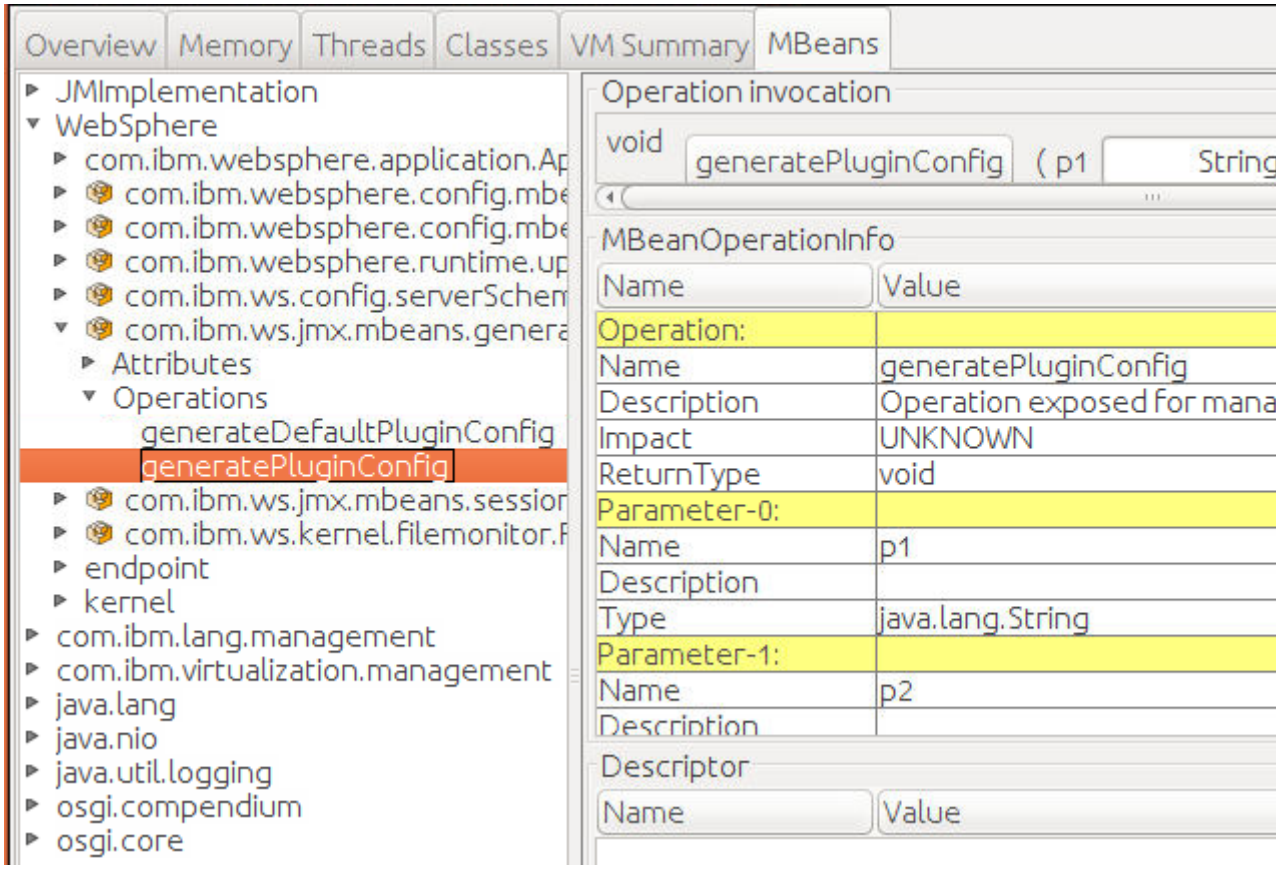
__ f. Click the **MBeans** tab.



__ g. In the navigation on the left, click to expand **WebSphere > com.ibm.ws.jmx.mbeans.generatePluginConfig > Operations**.

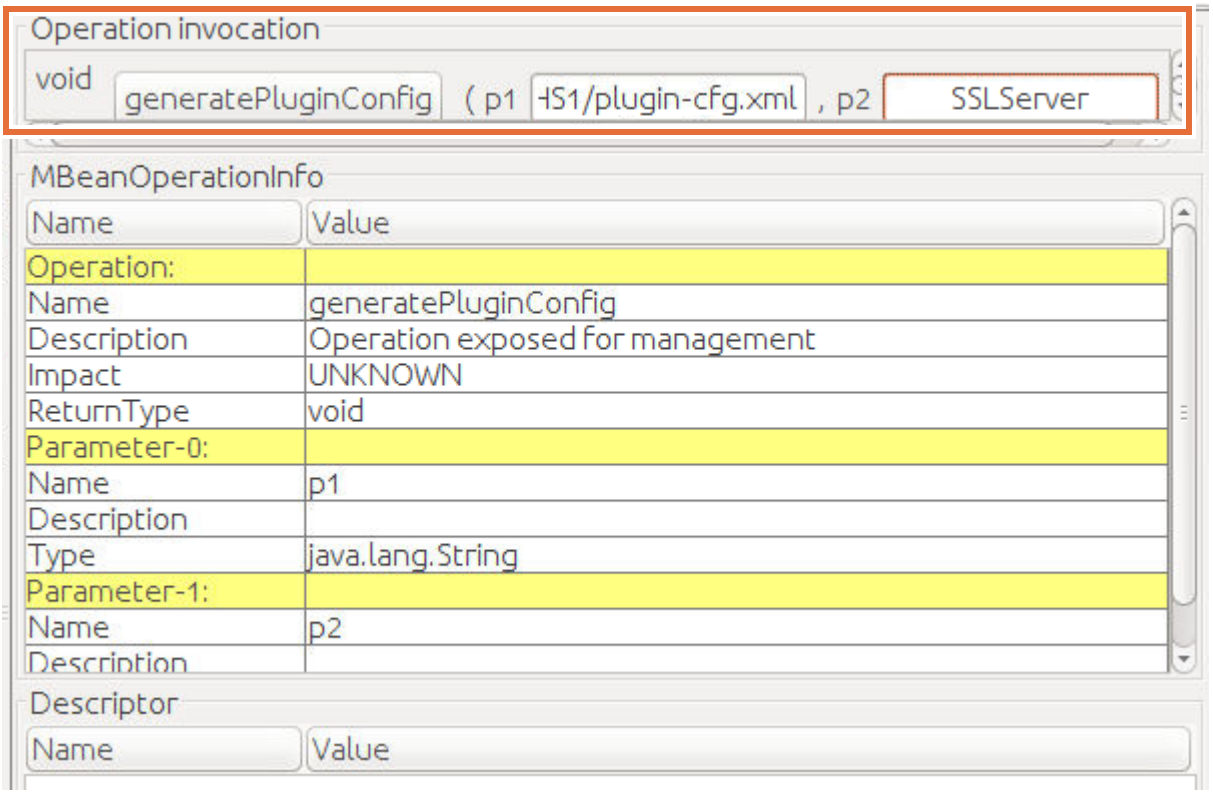


__ h. Under Operations, click **generatePluginConfig**. Information appears for generatePluginConfig on the **MBeans** tab.



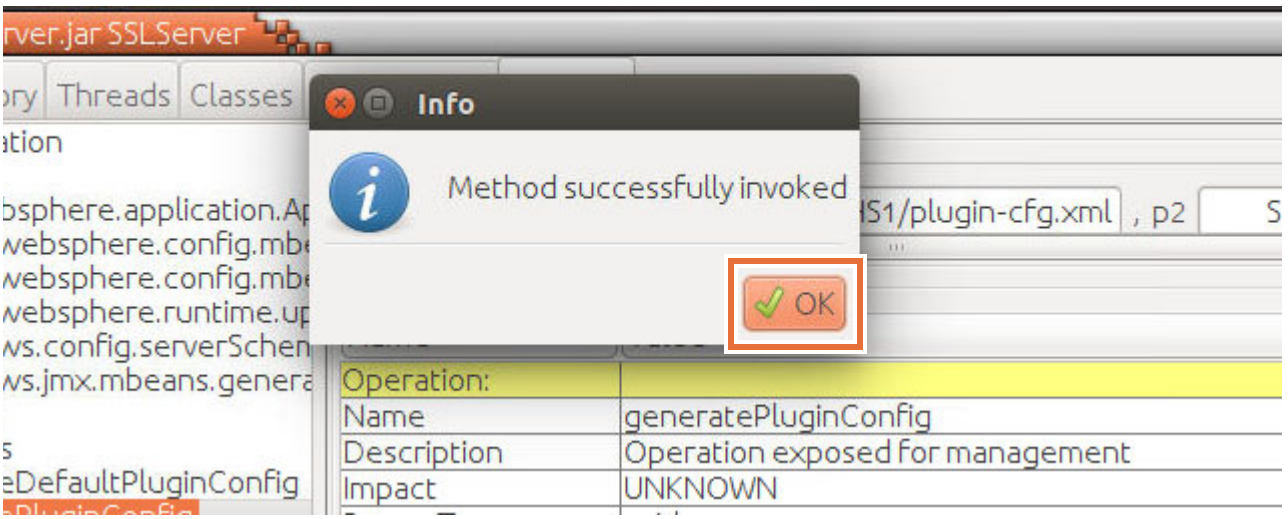
__ i. In the Operation invocation section, you can see that the `generatePluginConfig` operation takes two arguments. The required location of the plug-in configuration file and the name of the Liberty server name. Enter the following information:

- In the **p1** field, enter: `/opt/IBM/WebSphere/Plugins/config/IHS1`
- In the **p2** field, enter: `SSLServer`



__ j. Verify the changes. When completed, click **generatePluginConfig**.

__ k. In the Info window, click **OK**.



__ l. Exit jConsole. Click **Connection > Exit**.

__ 6. The updated `plugin-cfg.xml` file appears in the working directory for the server. You must copy it to the correction location that is used by the plug-in.

__ a. Go to the terminal window.

__ b. Go to the `/opt/wlp/usr/servers/SSLServer` directory.

__ c. List the contents directory. You can see that `plugin-cfg.xml` file appears.

__ d. To copy the file to the directory for the plug-in, enter the following command:

```
cp plugin-cfg.xml /opt/IBM/WebSphere/Plugins/config/IHS1/
```

__ e. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ f. Create the `logs/SSLServer` directory. To create the directory, enter the following commands:

```
mkdir logs
mkdir logs/SSLServer
```

__ 7. Examine the `plugin-cfg.xml` file.

__ a. Open the `plugin-cfg.xml` file and examine the settings. Open the file by using an editor such as vi or gedit. You can examine the file with any text editor, but be careful not to modify it now.

__ b. Go to the **LogLevel** directive. Notice that it specifies a log file that is named `http_plugin.log` in the `logs/SSLServer` directory.

plugin-cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?><!--HTTP server plugin config
file for SSLServer generated on 2016.09.28 at 10:47:41 EDT-->
<Config ASDisableNagle="false" AcceptAllContent="false"
AppServerPortPreference="HostHeader" ChunkedResponse="false"
FIPSEnable="false" IISDisableNagle="false" IISPluginPriority="High"
IgnoreDNSFailures="false" RefreshInterval="60" ResponseChunkSize="64"
SSLConsolidate="false" TrustedProxyEnable="false"
VHostMatchingCompat="false">
   <Log LogLevel="Error" Name="/opt/IBM/WebSphere/Plugins/config/IHS1/
logs/SSLServer/http_plugin.log"/>
   <Property Name="ESIEnable" Value="true"/>
   <Property Name="ESIMaxCacheSize" Value="1024"/>
   <Property Name="ESIInvalidationMonitor" Value="false"/>
   <Property Name="ESIEnableToPassCookies" Value="false"/>
   <Property Name="PluginInstallRoot" Value="/opt/IBM/WebSphere/
Plugins/config/IHS1"/>
<!-- Configuration generated using
httpEndpointRef=defaultHttpEndpoint-->
<!-- No virtual hosts are configured to accept requests from the
webserver http port (*:80).
```

__ c. Go to the **Route** directive at the bottom of the file. Notice that the directive ties the **VirtualHostGroup**, **UriGroup**, and **ServerCluster** elements together.

plugin-cfg.xml

```
      <Server CloneID="56db45ed-b35a-48a6-a161-475ca8037299"
ConnectTimeout="5" ExtendedHandshake="false" MaxConnections="-1"
Name="default_node_SSLServer" ServerIOTimeout="900"
WaitForContinue="false">
         <Transport Hostname="localhost" Port="9080" Protocol="http"/>
         <Transport Hostname="localhost" Port="9443" Protocol="https">
            <Property Name="keyring" Value="keyring.kdb"/>
            <Property Name="stashfile" Value="keyring.sth"/>
            <Property Name="certLabel" Value="LibertyCert"/>
         </Transport>
      </Server>
      <PrimaryServers>
         <Server Name="default_node_SSLServer"/>
      </PrimaryServers>
   </ServerCluster>
   <UriGroup Name="default_host_SSLServer_default_node_Cluster_URIs">
      <Uri AffinityCookie="JSESSIONID"
AffinityURLIdentifier="jsessionid" Name="/SimpleSSL/*"/>
   </UriGroup>
   <Route ServerCluster="SSLServer_default_node_Cluster"
UriGroup="default_host_SSLServer_default_node_Cluster_URIs"
VirtualHostGroup="default_host"/>
</Config>
```

The **Route** directive instructs the plug-in to forward requests for URLs that match the UriGroup, or `/SimpleSSL/*`, and requests that are sent to hosts in the VirtualHostGroup. In this case, any requests that arrive on port 9180 or port 8443. The requests should be forwarded to a WebSphere server in ServerCluster, which is the single server that listens on ports 9080 and 9443, the Liberty server. The plug-in causes the web server to forward any requests that belong to the SimpleSSL application to Liberty, and it handles all other requests itself.

__ d. When completed, close the file.

Information

If you had an actual cluster instead of just one Liberty server, then the `plugin-cfg.xml` file might also include load balancing directives and other directives that are related to handling Liberty servers that are temporarily unavailable.

The `plugin-cfg.xml` file is directly tied to your Liberty topology. Any changes that you make to the topology, you must update the `plugin-cfg.xml` file. If you add more applications to your Liberty server, you must regenerate a new plug-in configuration file by using jConsole to access the new applications through the web server. The new `plugin-cfg.xml` file would include your new applications in the UriGroup, so the plug-in would forward those requests to Liberty, and requests for the SimpleSSL application. With a Liberty cluster, you must regenerate a new plug-in

configuration file whenever you add or remove cluster members to update the ServerCluster directives.

Part 5: Testing the plug-in configuration

__ 1. Restart the web server.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ c. Enter the following command to start the HTTP Server:

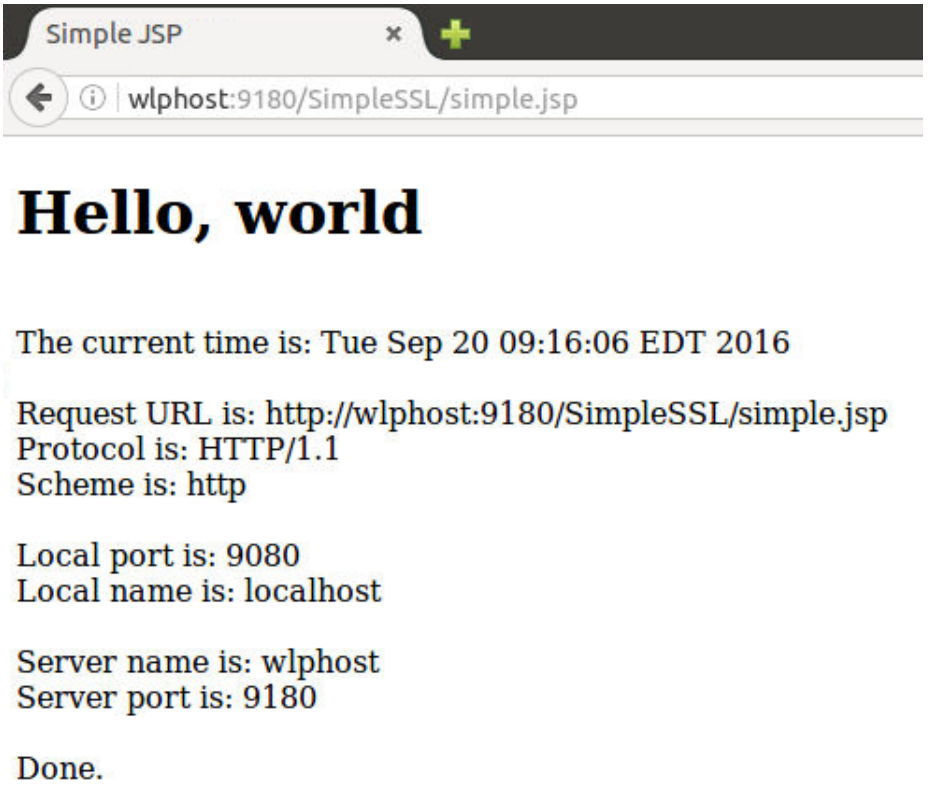
```
./apachectl start
```

__ 2. Test the configuration.

__ a. Open a Firefox web browser.

__ b. Test the configuration by sending an HTTP request to the web server and verify that the plug-in forwards the request to the Liberty server. To test, enter the following URL:

```
http://wlphost:9180/SimpleSSL/simple.jsp
```



This output shows that the local port is 9080, the Liberty server, and the server port is 9180, the web server.

__ c. Next, test an HTTPS request. Enter the following URL:

```
https://wlphost:8443/SimpleSSL/simple.jsp
```

__ d. In the Your connection is not secure pane, click **Advanced**. Click **Add Exception**.

__ e. Clear the **Permanently store this exception** check box.

__ f. Click **Confirm Security Exception**.

__ g. The output shows an internal server error. This error is because you need to configure SSL between the plug-in and the Liberty server.

__ h. Close the browser window.

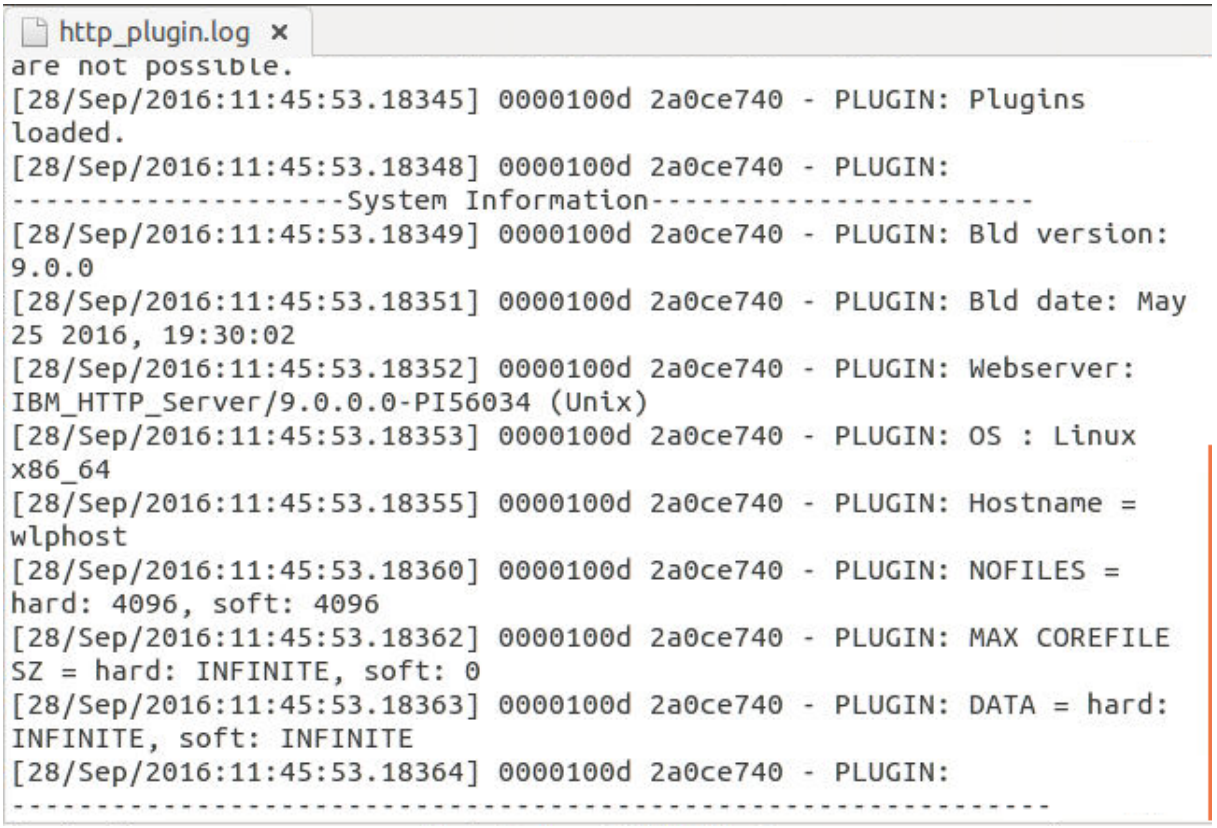
__ 3. Examine the log file.

__ a. Go to a terminal window.

__ b. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1/logs/SSLServer` directory.

__ c. Open the `http_plugin.log` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. You can see security errors when the server starts, because of the incomplete SSL configuration between the web server and the Liberty server, SSLServer. However, you should see details that the plug-in started successfully without SSL. You can see information that the plug-in loaded and specifics on the plug-in.



__ e. When completed, close the file.

Part 6: Configuring SSL between the plug-in and Liberty

You have a simple JSP running in a web application on a Liberty server. You can access that JSP directly by using either HTTP or HTTPS. You have an IBM HTTP Server and you can access its

documents by using either HTTP or HTTPS. Additionally, you configured the web server plug-in so that you can access Liberty through the web server, while you use only HTTP and not HTTPS.

In this part of the exercise, you configure SSL between the plug-in and Liberty, so you can access Liberty by using HTTPS through the web server.

__ 1. Modify the plug-in configuration.

__ a. Go to the terminal window.

__ b. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ c. Open the `plugin-cfg.xml` file and examine the settings. Open the file by using an editor such as vi or gedit.

__ d. Go to the Transport directive.

__ e. Examine the details for both Transport and Property.

plugin-cfg.xml

```
      <Server CloneID="56dD45ed-D35a-48a6-a161-475ca8037299"
ConnectTimeout="5" ExtendedHandshake="false" MaxConnections="-1"
Name="default_node_SSLServer" ServerIOTimeout="900"
WaitForContinue="false">
         <Transport Hostname="localhost" Port="9080" Protocol="http"/>
         <Transport Hostname="localhost" Port="9443" Protocol="https">
            <Property Name="keyring" Value="keyring.kdb"/>
            <Property Name="stashfile" Value="keyring.sth"/>
            <Property Name="certLabel" Value="LibertyCert"/>
         </Transport>
      </Server>
      <PrimaryServers>
         <Server Name="default_node_SSLServer"/>
      </PrimaryServers>
```

The `plugin-cfg.xml` file is already prepared for SSL. You can see that it contains directives that specify a key ring for the plug-in to use when contacting Liberty by using SSL. However, you need to create the proper `keyring.kdb` and `keyring.sth` files.

__ f. Change the value for the name of the key ring and stash file. Also, use the absolute path for each. To define the path, make the following changes:

```
<Property Name="keyring"
Value="{/opt/IBM/WebSphere/Plugins/config/IHS1/plugin-key.kdb"/>
<Property Name="stashfile"
Value="{/opt/IBM/WebSphere/Plugins/config/IHS1/plugin-key.sth"/>
```

Information

The key database file (`.kdb`) contains the signing certificate for the Liberty server, and the stash file (`.sth`) holds the password for the `.kdb` file.

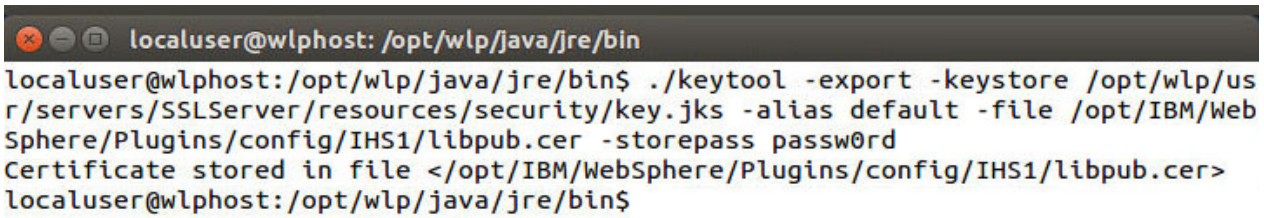
__ g. Save and close the file.

__ 2. Configure the certificates.

__ a. Go to the `/opt/wlp/java/jre/bin` directory.

__ b. Extract the signing certificate for the Liberty server from its keyfile. To extract the signing certificate, enter the following command:

```
./keytool -export -keystore
/opt/wlp/usr/servers/SSLServer/resources/security/key.jks -alias default
-file /opt/IBM/WebSphere/Plugins/config/IHS1/libpub.cer -storepass
passw0rd
```

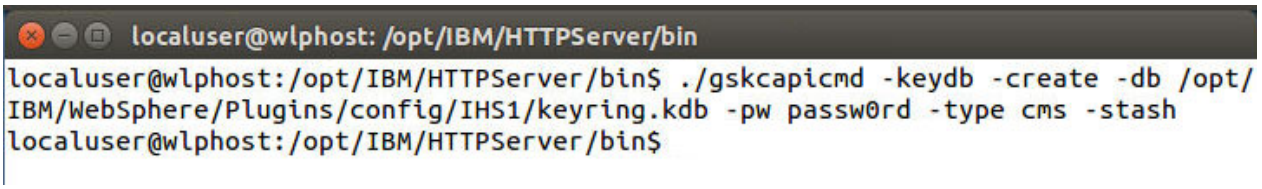


You can see from the output that this produces a file that is named `libpub.cer`.

__ c. Go to the `/opt/IBM/HTTPServer/bin` directory.

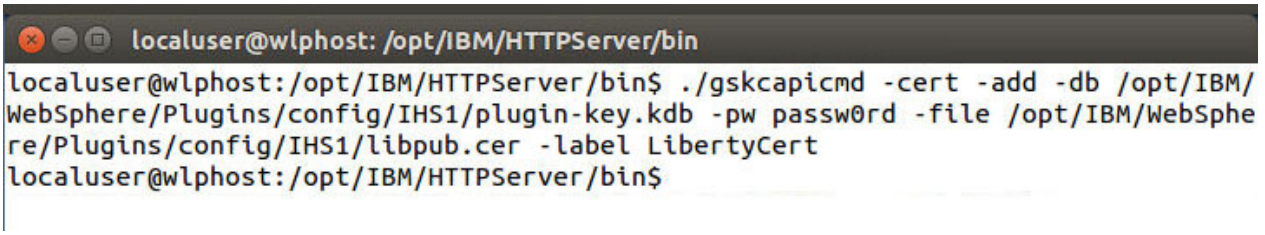
__ d. Next, create a separate keystore for the plug-in, then add the certificate that is contained in `libpub.cer` to the new keystore for the plug-in.

```
./gskcapicmd -keydb -create -db
/opt/IBM/WebSphere/Plugins/config/IHS1/plugin-key.kdb -pw passw0rd -type
cms -stash
```



__ e. Add the certificate from `libpub.cer` to the keystore with label `LibertyCert`. To add the certificate, enter the following command:

```
./gskcapicmd -cert -add -db
/opt/IBM/WebSphere/Plugins/config/IHS1/plugin-key.kdb -pw passw0rd -file
/opt/IBM/WebSphere/Plugins/config/IHS1/libpub.cer -label LibertyCert
```



__ f. Go to the `/opt/IBM/WebSphere/Plugins/config/IHS1` directory.

__ g. List the contents of the directory. There are several files in with the name of `plugin-key` and different suffixes.

__ h. Delete the `libpub.cer` file, as it is no longer needed.

__ 3. Restart the web server.

__ a. Go to the `/opt/IBM/HTTPServer/bin` directory.

__ b. Enter the following command to stop the HTTP Server:

```
./apachectl stop
```

__ c. Enter the following command to start the HTTP Server:

```
./apachectl start
```

__ 4. Test the configuration.

__ a. Open a Firefox web browser.

__ b. Test the configuration by sending an HTTPS request to the web server and verify that the plug-in forwards the request to the Liberty server. To test, enter the following URL:

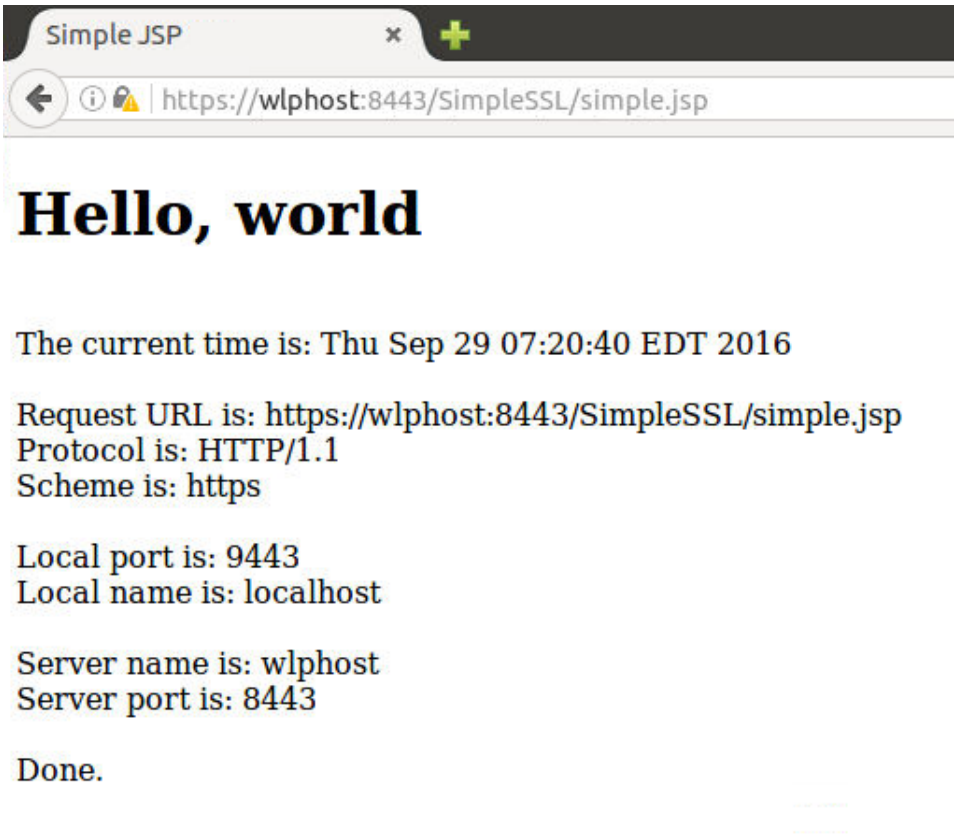
```
https://wlphost:8443/SimpleSSL/simple.jsp
```

__ c. In the Your connection is not secure pane, click **Advanced**. Click **Add Exception**.

__ d. Clear the **Permanently store this exception** check box.

__ e. Click **Confirm Security Exception**.

__ f. You can see the details for Hello, world.



This output shows that the local port is 9443, the Liberty server, and the server port is 8443, the web server. The entire request is using SSL.

Information

This type of configuration, by using a web server as a front end for a Liberty server, is useful in several different situations. You can transparently serve static content (unchanging HTML files like the “It works!” page) from a web server, and serve dynamic content like JSPs and servlets from Liberty. In more complex environments, the plug-in can be configured to do load balancing among multiple clustered Liberty servers, avoiding any Liberty servers that might be down, for high availability and better scalability.

End of exercise

Exercise review and wrap-up

In this exercise, you configured Liberty, the IBM HTTP Server, and the plug-in that connects them to use SSL. You secured the communication from the browser, to the web server, plug-in, and onto Liberty.

IBM Training