Course 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.

# Course description

## Administering WebSphere Application Server Liberty Profile

## Duration: 2 days

## Purpose

This course teaches you the skills that are needed to manage Liberty servers and collectives.

The course is designed for application server administrators. You learn how to use the graphical Admin Center and the command line scripting to manage servers from a collective controller. The course also covers how to deploy a cluster of packaged servers for Liberty runtimes, view the deployment environment, and view basic performance metrics.

You learn how to use the Dynamic Routing feature of Liberty to enable routing of HTTP requests to collective members. You also configure the auto-scaling and health management features for Liberty.

Finally, you learn how to secure Liberty and enable SSL communication in Liberty.

## Audience

This course is designed for administrators of IBM WebSphere Application Server Liberty Profile.

## Prerequisites

Before taking this course, you should have a general knowledge of:

- Java Platform, Enterprise Edition (Java EE)
- Administering web servers and application servers
- The Ubuntu Linux operating system

## Objectives

- Describe the WebSphere Liberty Profile architecture
- Create a Liberty profile server
- Use the Admin Center to manage Liberty servers
- Deploy clusters of Liberty servers
- Use the collective controller
- Use Jython scripts to administer Liberty servers
- Configure Dynamic Routing
- Configure the auto scaling feature and define auto scaling policies

- Configure SSL communication in Liberty
- Use the IBM HTTP and web server plug-in with Liberty servers

# Agenda

**Note**

The following unit and exercise durations are estimates, and might not reflect every class experience.

## Day 1

(00:15) Course introduction
(01:00) Unit 1. Introduction to Liberty administration and runtime architecture
(01:30) Unit 2. Multi-server management
(01:15) Exercise 1. Managing Liberty collectives with the Admin Center
(00:45) Unit 3. Administration and application deployment with scripting
(01:30) Exercise 2. WebSphere Liberty administration by using Jython Scripts

## Day 2

(00:45) Unit 4. Dynamic Routing
(01:15) Exercise 3. Dynamic Routing
(00:45) Unit 5. Auto-scaling in Liberty
(01:15) Exercise 4. Auto-scaling
(00:30) Unit 6. Securing Liberty
(01:15) Exercise 5. Using the IBM HTTP Server with SSL to a Liberty server
(00:15) Unit 7. Course summary

# Unit 1. Introduction to Liberty administration and runtime architecture

## Estimated time

01:00

## Overview

This unit describes the configurable architecture of Liberty.

## How you will check your progress

- Review questions

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

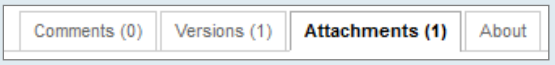
IBM Training

# How to check online for course material updates


**Note:** If your classroom does not have Internet access, ask your instructor for more information.

## Instructions

1. Enter this URL in your browser: **ibm.biz**/CloudEduCourses
2. Find the product category for your course, and click the link to view all products and courses.
3. Find your course in the course list and then click the link.
4. The wiki page displays information for the course. If there a course corrections document is available, this page is where it is found.
5. If you want to download an attachment, such as a course corrections document, click the **Attachments** tab at the bottom of the page.




6. To save the file to your computer, click the document link and follow the prompts.




*Figure 1-1. How to check online for course material updates*

IBM Training

## Unit objectives

- Describe Liberty
- Describe the composable architecture of Liberty
- Describe the Liberty configuration file
- Describe the continuous delivery of new Liberty function
- Describe the flexible deployment of applications
- Describe the dynamic nature of the configuration
- Describe the manual administration of Liberty servers and features




*Figure 1-2. Unit objectives*

# 1.1. Liberty overview

IBM Training

# Liberty overview




*Figure 1-3. Liberty overview*

IBM Training

## Topics

- Liberty overview
- Commonly used server commands




*Figure 1-4. Topics*

IBM Training

## What is Liberty?

- Lightweight, flexible Java EE runtime within the WebSphere Application Server product set
- Matching Liberty edition for every edition of WebSphere Application Server
  - WebSphere Application Server Liberty (“Base”)
  - WebSphere App Server Network Deployment Liberty (“ND”)
  - WebSphere App Server Liberty z/OS
- Including its own, low-end edition
  - WebSphere Application Server Liberty Core
- Liberty consists of a kernel and a set of pluggable features
  - Each product edition has a different set of features




*Figure 1-5. What is Liberty?*

Liberty is a lightweight, modular profile of IBM WebSphere Application Server. Liberty provides a dynamic, flexible runtime 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. Within the development environment, Liberty supports the same platforms as the full application server, plus the Mac OS X operating system. 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.

Because a Liberty server is lightweight, it can be packaged easily with applications in a compressed file. This package can be stored, distributed to colleagues, and used to deploy the application to a different location or to another system. It can even be embedded in your own product distribution.

Starting with WebSphere Application Server V8.5, Liberty is included with each package. Liberty is highly composable, where you have many features that are installed but you can configure only the features that you need. You can design your server configuration to match the needs of your applications.

Liberty is also available as a stand-alone offering, called WebSphere Application Server Liberty Core. Each package, except for Liberty Core and Community Edition, includes both the WebSphere Classic application server and a Liberty application server. The features available for each runtime, for example programming model support, vary among the different packaging options.

The Liberty profile supports a subset of the following parts of the full WebSphere Application Server programming model: Web applications, OSGi applications, and the Java Persistence API (JPA).

IBM Training

## Composable feature sets

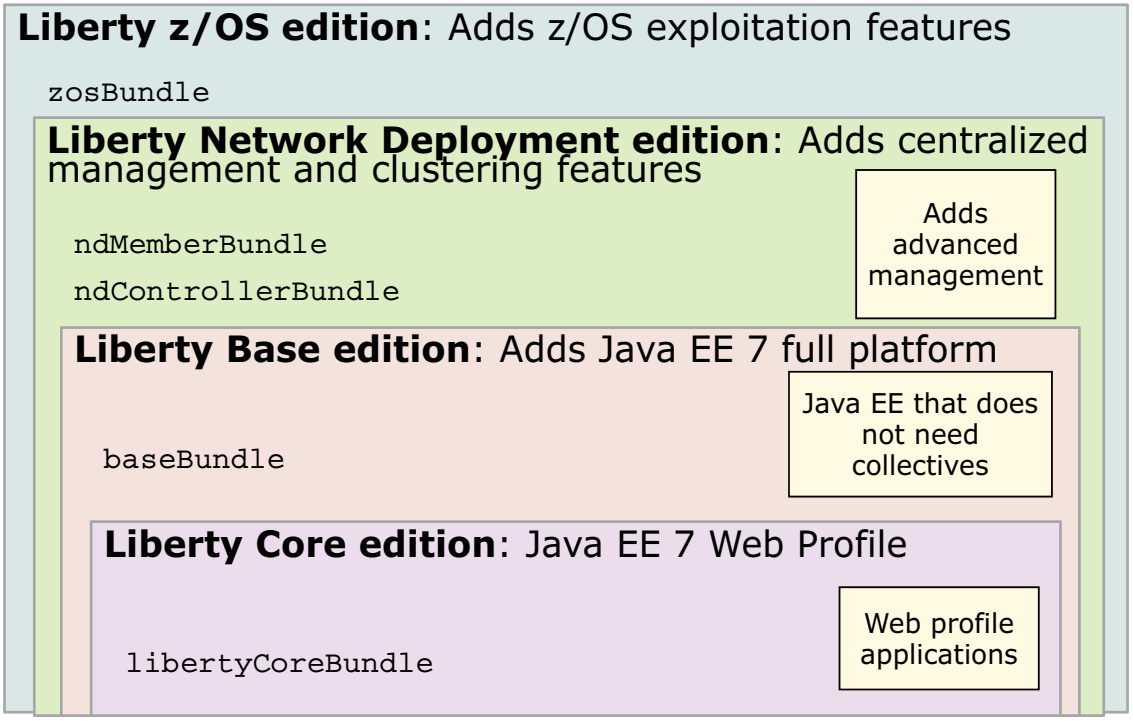






*Figure 1-6. Composable feature sets*

Because different application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging editions.

WebSphere Application Server provides two runtime profiles: WebSphere Application Server Classic and Liberty. The runtime that has always been available with the WebSphere Application Server is referred to as WebSphere Application Server Classic, also known as the full profile. The application serving runtime, provided by WebSphere Application Server Classic, is composed of a wide spectrum of components that are always available in the application server.

The Liberty profile included in every edition of WebSphere Application Server. Each higher edition has a larger set of features.

WebSphere classic is present in all editions except Liberty Core edition.

If you buy WebSphere Application Server (base) or WebSphere Application Server Network Deployment, you can use both liberty and traditional WebSphere, the same license covers both products.

It is important to understand what you get when you purchase a specific edition and what installation options you must select from.

To install all features that apply to a Liberty edition, you can install a feature bundle add-on.

- WebSphere Application Server Liberty Core: libertyCoreBundle
- WebSphere Application Server (base) Liberty: baseBundle
- WebSphere Application Server Network Deployment Liberty: ndMemberBundle for collective member servers and ndControllerBundle for collective controllers
- WebSphere Application Server for z/OS Liberty: zosBundle

IBM Training

## Reasons for choosing Liberty

- Reduce costs, simplify, and become Cloud-ready
- Development efficiencies
  - Small, simple, up and running quickly
  - Fast restarts, few restarts
- Flexible licensing
  - Choose Core or Base editions based on API needs
  - Add Network Deployment for central management and advanced clustering
- Flexible administration
  - Small compressed file deployment
  - Simple configuration files, dynamic changes
  - Product management scripts
  - Zero migration architecture
  - Self-tuning thread pools, dynamic routing, auto scaling
- Great performance
  - Similar to traditional WebSphere Application Server on distributed platforms




*Figure 1-7. Reasons for choosing Liberty*

The slide provides some reasons why a customer might choose Liberty as their application server runtime.

Benefits include reducing operational costs, simplifying the development and deployment processes, and becoming cloud-ready.

Liberty: A modern application server for hosting Java EE and OSGi applications during development, test, and production.

- Starts and stops quickly, typically in a few seconds.
- Requires little memory and disk space.
- Simple to configure with an XML file. Include only the capabilities your applications require.
- Responds dynamically to application updates and server configuration changes.
- Built by using IBM technologies and open source software packages from organizations such as Apache, Eclipse, and OW2.

(The OW2 Consortium is an open source community that promotes the development of open source middleware, generic business applications, and cloud computing).

Liberty is a highly composable and dynamic runtime environment. It only activates features that you configure, so the memory footprint is small. For example, a benchmark application that is named

TradeLite runs under 64 MB of Java heap size. Which means you can run many application instances per machine, which can diminish the megabyte-hours that you pay for.

Liberty starts and stops quickly because of the lightness of the runtime size and memory footprint. It can start in just a few seconds. Liberty promotes application elasticity, enabling applications to rapidly scale-up and scale-down as the workload changes.

Liberty also provides a no-charge and no-support option for web-centric applications for use in small test and production environments, which includes both on-premises or in the cloud. This use is restricted to a total of 2 GB of Java virtual machine (JVM) heap size across all instances of application servers for the licensee. IBM also provides an in-place option to upgrade from a no-charge, no-support to other WebSphere Application Server package offerings. Liberty is flexible with license upgrades and there is a simple process to upgrade your Liberty license. For example, if you downloaded and are using the Liberty development runtime, which includes a development license. And you want to upgrade this license to a full production license. Perhaps you want clustering and auto scaling capabilities in your runtime, which requires the Network Deployment license. To upgrade, go to Passport Advantage and download the license that you need. The license is a JAR file that you apply to your production-ready machine. In this simple process, you were able to upgrade easily from a development license to a fully supported production Network Deployment license.

# Liberty Repository

- Repository of Liberty product assets available for download

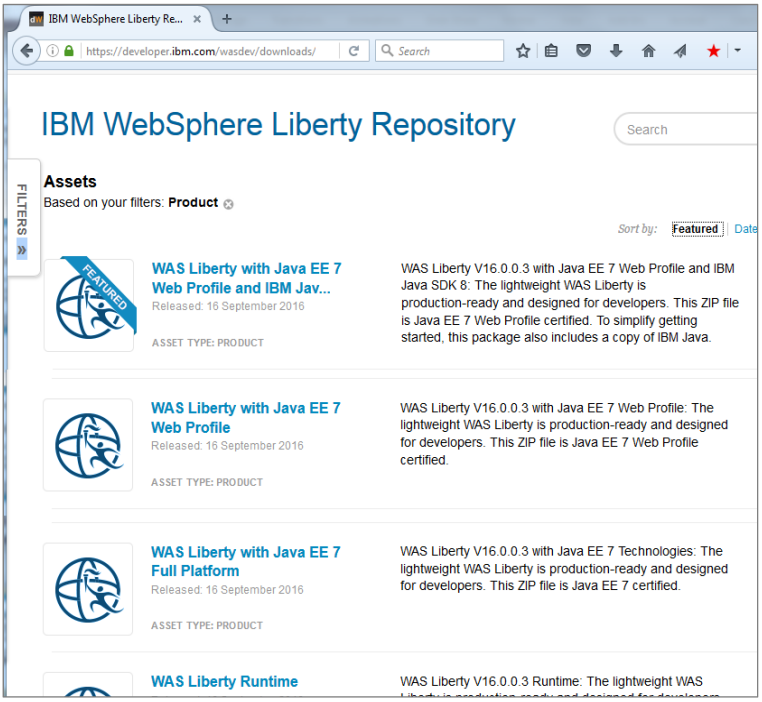






*Figure 1-8. Liberty Repository*

The Liberty Repository at https://developer.ibm.com/wasdev/downloads/ is the site where you can download Liberty assets from to get started with Liberty.

The rapid evolution and adoption of cloud, mobile, and social media technologies are driving the demand for delivering applications faster and more frequently. WebSphere Application Server is now delivering features for Liberty on a continual basis by using the Liberty Repository. 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. You can use the Liberty Repository to easily extend or enhance your Liberty-based applications. The optional, production-ready features can be quickly and easily added to an existing Liberty installation. Choose the features that you want and then install the features to the applicable product service level. The features that you add inherit the same support of your existing installation.

IBM Training

## Composable installation

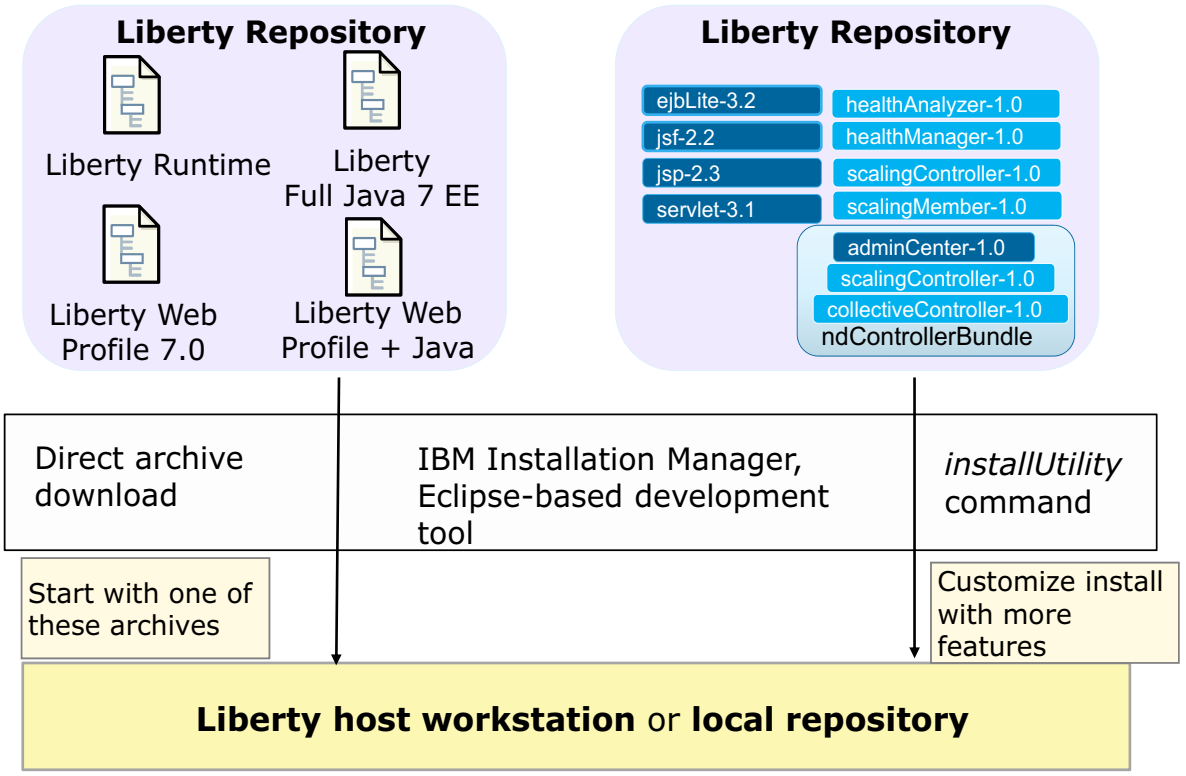






*Figure 1-9. Composable installation*

WebSphere Liberty starter sets include the Liberty kernel plus the most popular sets of features. Archives that include Java are also available on the download repository.

Manually download and decompress the file, or download and install by using IBM Installation Manager or an Eclipse-based development tool, such as the WebSphere Developer Toolkit.

More features are available separately, and in convenience bundles, for specific editions (core, base), or for special purposes (ndMember, ndController).

These features can be downloaded and installed manually by using the `installUtility` command, or by using IBM Installation Manager or the WebSphere Developer Toolkit.

The archives can be installed directly onto the machine where they are to be used, or into a local repository for distribution within the company firewall.

---

**Example**

If you install the Liberty webProfile-7.0 feature, these features are enabled:

appSecurity-2.0: Application Security 2.0

beanValidation-1.1: Bean Validation 1.1

cdi-1.2: Contexts and Dependency Injection 1.2

ejbLite-3.2: Enterprise JavaBeans Lite 3.2

el-3.0: Expression Language 3.0

jaxrs-2.0: Java RESTful Services 2.0

jdbc-4.1: Java Database Connectivity 4.1

jndi-1.0: Java Naming and Directory Interface

jpa-2.1: Java Persistence API 2.1

jsf-2.2: JavaServer Faces 2.2

jsonp-1.0: JavaScript Object Notation Processing

jsp-2.3: JavaServer Pages 2.3

managedBeans-1.0: Java EE Managed Bean 1.0

servlet-3.1: Java Servlets 3.1

websocket-1.1: Java WebSocket 1.1

IBM Training

## Lightweight download and install

- Download the required archive from http://wasdev.net/repo
- You can install a Liberty profile runtime environment by extracting a compressed file
  - A supported Java runtime environment (JRE) must be installed separately
- Liberty with Java EE 7 Web Profile and IBM Java SDK 8 are also available from the Liberty Repository
  - Single compressed file with Liberty runtime and IBM Java SDK
- Installing extra features
  - The `installUtility` (located in the `wlp/bin` folder) can be used to install extra features from the Liberty Repository
  - Install individual features such as `adminCenter-1.0`, `servlet-3.0`, or a feature bundle such as ndControllerBundle (includes `adminCenter-1.0` and `collectiveController-1.0`)




*Figure 1-10. Lightweight download and install*

A free download of the Liberty runtime is available on the IBM developerWorks WASdev site at http://developer.ibm.com/wasdev/downloads/.

Accept the license terms and download the file.

You can install the Liberty runtime on any supported Java 7+ runtime.

IBM WebSphere Liberty with Java EE 7 Web Profile and IBM Java SDK 8 are also available from the WebSphere Liberty Repository as a single downloadable compressed (ZIP) file.

Because Liberty does not include a Java runtime environment (JRE), you must install a compliant Java implementation (JRE or SDK) beforehand.

The installUtility (located in the `wlp/bin` folder) can be used to install more features from the Liberty Repository such as adminCenter-1.0, servlet-3.0, collectiveController-1.0, or a feature bundle such as ndControllerBundle (includes adminCenter-1.0 and collectiveController-1.0).


**Example**

Running the `installUtility` command installs the Liberty Network Deployment components:

```
<wlp_root>/bin/installUtility install ndMemberBundle –acceptLicense
<wlp_root>/bin/installUtility install ndControllerBundle --acceptLicense
```

IBM Training

## Commands to install features

Use the **featureManager** command to obtain details of all the features that are installed

- Run the featureManager command from `<wlp_root>/bin`
- For example, run the following command to create an XML file that contains a detailed report of all your features
  - `featureManager featureList myfeatures.xml`

Use the **installUtility** to install, find, or download assets from multiple repositories

- Format:
  `installUtility` *`action`* `[options]`




*Figure 1-11. Commands to install features*

You can use the `featureManager` command to install Liberty Repository features in your Liberty environment. Use the `featureManager` command to obtain details of all the features that are installed.

The `featureManager` find, install, and uninstall actions are stabilized.

Use the `installUtility` command instead of the `featureManager` command. With the `installUtility` command, you can manage more asset types and install, find, or download assets from multiple repositories.

After you install Liberty, you can install assets by running the `installUtility` command. The `installUtility` command automatically installs asset dependencies.

By default, the `installUtility` command is configured to install assets only from the Liberty Repository. If you want to install assets from a local directory-based repository or an instance of the Liberty Asset Repository Service, you must configure these repositories in the `repositories.properties` file.

IBM Training

## Continuous delivery on new function

- Beta drivers every month
- Generally available features are delivered regularly through the Liberty Repository
- Quarterly fix packs

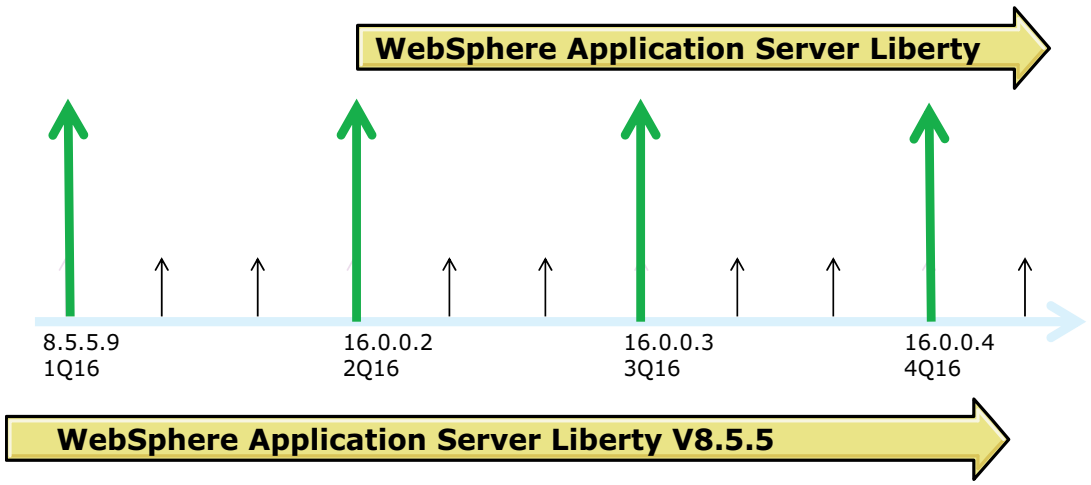






*Figure 1-12. Continuous delivery on new function*

Liberty has moved to a continuous delivery model. The continuous delivery model provides new optionally installable features and functions, which can be added to an existing Liberty installation at the latest 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.

Beta drivers are delivered every month so you can preview what is coming and provide feedback. New, supported features are available in the online repository at wasdev.net to add to your existing installation as and when you need them. Fixpacks are made available each quarter.

The first version delivery of Liberty is the second fix pack of 2016 and is numbered 16.0.0.2.

IBM is emphasizing the version-less delivery stream by using a numbering scheme based on year and fix pack number. Therefore, you might see that the name of the Liberty product does not include the version number for any Liberty version with fix pack 16.0.0.2 or greater.

## Composable server instances

- You control which features are loaded into each server instance
- One required XML configuration file
  - `<wlp_root>/usr/servers/<serverName>/server.xml`

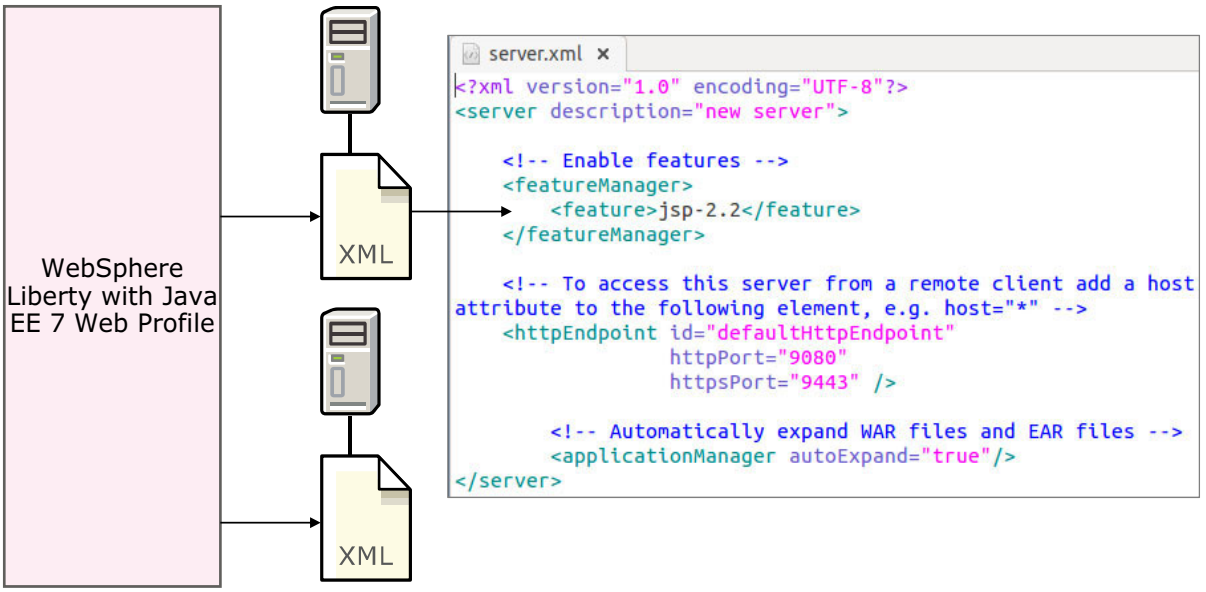






*Figure 1-13. Composable server instances*

A major aspect of the simplicity of liberty is the configuration files. Instead of dozens of separate XML files over many directories, you can configure a server in a single XML file.

The `server create` command generates a simple working `server.xml` and `default` folders in the `<wlp_root>/usr/servers/<serverName>/` directory, where `<wlp_root>` is the installation directory of the Liberty runtime.

Use the server configuration to control which features are loaded into a given server instance at a fine-grained level, so you get exactly the function you want and no more.

You specify which features you want the server to run by adding them to the `<featureManager>` stanza in the `server.xml` file.

You can use the `server.xml` file to configure other parameters, such as the server listening ports and data sources.

IBM Training

## Simplified configuration: server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>adminCenter-1.0</feature>
    </featureManager>

    <logging traceSpecification=
        "webcontainer=all=enabled:*=info=enabled" />

    <webApplication id="static" location="static.war

    <include location="${server.config.dir}/collective-create-
       include.xml" />
</server>
```

Features control which capabilities (OSGi bundles) are installed in the server

Singleton configurations specify properties for a runtime service like logging

Instance configurations specify multiple resources such as application definitions

Any of this configuration might be put into a separate XML file and included in this master configuration file by using the **include** element




*Figure 1-14. Simplified configuration: server.xml*

In Liberty, configuration information is stored in XML files. In the simplest case, one XML file, `server.xml`, is used for all server configurations. This file can be manually updated with a text editor and when the changes are saved, the server is dynamically updated. No server restart is required. More XML configuration files and their contents can be accessed by using an include statement in the `server.xml` file.

The file can be exported, shared, and versioned. Specifying which feature capabilities are included in a server is simple, and part of the reason why Liberty is so easy to use in any cloud or virtualization container.

Features are the units of functions that control the parts of the runtime environment that are loaded into a Liberty server. Each Liberty server is configured by using a `server.xml` configuration file and features are specified in this file. Each feature has its own version identifier, so multiple versions of the same feature can run in the same server. Features can define programming models, administrative capabilities, security features, and more. The set of features differs between Liberty distributions which depends on the support that is provided with the edition.

Each feature has a version identifier. This identifier is provided so that multiple versions of the same feature can be used in subsequent releases. Some features include other features. For example, the jsp-2.2 feature includes the servlet-3.0 feature. This situation is because to run the JSP page, you need a web container. Similarly, the jsf-2.0 feature includes the jsp-2.2 feature. The feature

manager maps each feature name to a list of bundles that provide the feature. When a feature configuration is changed, the feature manager recalculates the list of required bundles. It stops and uninstalls those bundles that are no longer needed, and then installs and starts any additions. It also skips any features that are already loaded. All features are designed to cope with other features that are added or removed dynamically.

Features are included in a Liberty server in the following steps:

1. The OSGi Configuration Admin service reads the `server.xml` file and injects the feature configuration into the feature manager service.
2. The feature manager then maps each feature name to a list of bundles that provide the feature.
3. With all of the appropriate bundles ready, the feature manager installs and starts the features in the OSGi framework.

Although keeping all of the configuration settings for a server in a single file eases the complexity of server configuration, the file can grow to a substantial size. This is one reason to use the include syntax to move some of the configuration into other XML files. Furthermore, the included XML files can also include other configuration files.

The include syntax provides a flexible and powerful way to share all or part of a configuration between different servers on the same or even different host machines. You can control how the configuration is structured and which pieces are shared by which servers. Included XML files can be on the local file system or hosted in the network. The monitor service detects changes to the `server.xml` file and any of the included files.

## Manual feature management

- Use any text editor to manually update the `server.xml` file
- For example:
  - To add the Admin Center feature, add `adminCenter-1.0` to the `featureManager` element

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

- Updates to the server are dynamic
  - No need to restart the server




*Figure 1-15. Manual feature management*

Use any text editor to manually change features by adding or removing them from the `server.xml` file.

Updates to the server configuration are dynamic and take effect when the configuration changes are saved. The feature manager also responds to configuration changes by dynamically adding and removing features while the server is running.

IBM Training

# Dynamic enablement of feature sets

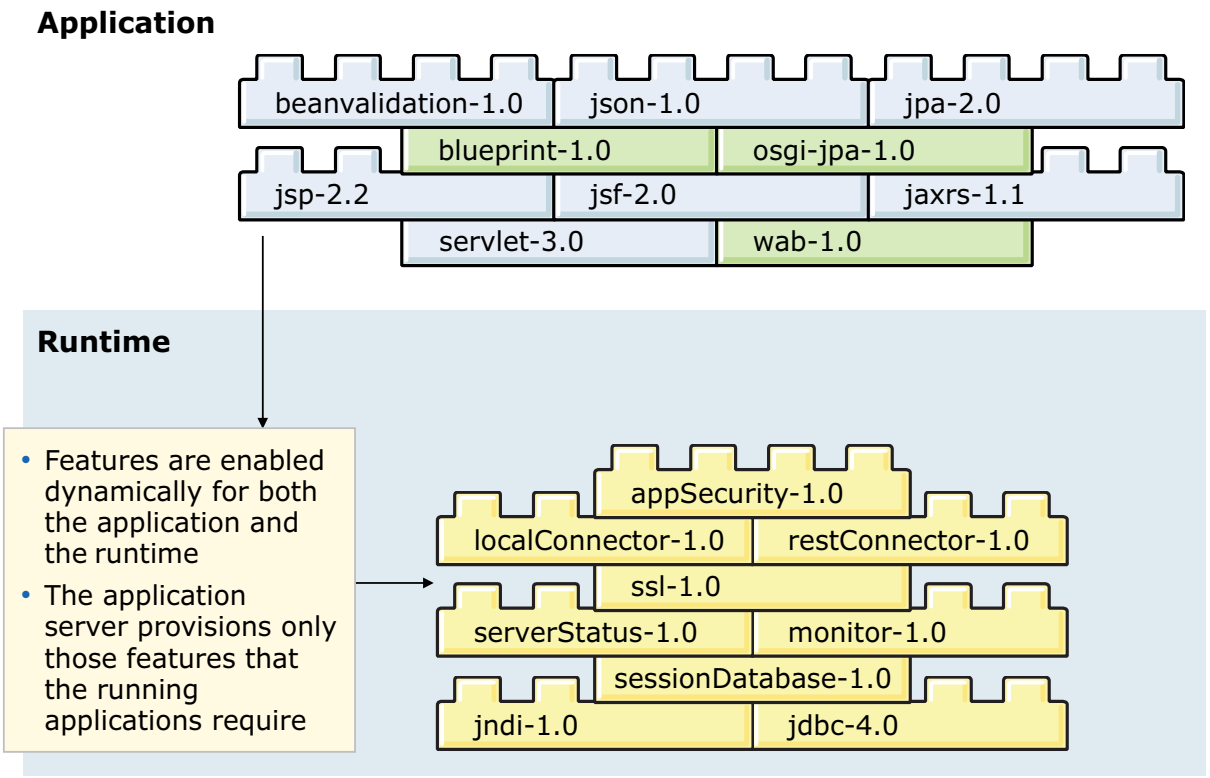






*Figure 1-16. Dynamic enablement of feature sets*

The Liberty profile provides an application server runtime environment that is a highly composable, fast to start, and dynamic. A highly composable system is defined as: “A system that provides components that can be selected and assembled in different combinations to provide specific application requirements.”

Features are enabled dynamically to the Liberty Core for both the application and the runtime. The application server provisions only those features that the running applications require.

Liberty profile server JVMs start only those features that you add to its `server.xml` file. By default, a server contains only the jsp- 2.2 feature to support servlet and JSP applications. You use the Liberty profile feature manager to add the features that you need.

The building blocks shown in the graphic on this slide show some of the Liberty profile features that can be defined for the feature manager. These features include, JPA, JSP, servlet, application security, and a remote JMX connector. However, several other features that might be configured for a particular server. In addition to the features, you can add HTTP port definitions for the HTTP transport, and application definitions for the application manager.

The graphics on this slide show two example feature sets for a Liberty Profile server. All features are dynamically activated when they are saved to the configuration file of the server. No server restart is required. The application features show services that applications require and that run on the server such as bean validation, osgi-jpa, and wab (web application bundle). The runtime

features show services that the server runtime itself requires such as serverStatus, monitor, localConnector, restConnector, and ssl.

## Application deployment

- Applications are deployed by using:
  - Monitored directory (`dropins`)
  - Configuration (`server.xml`)
  - Developer tools

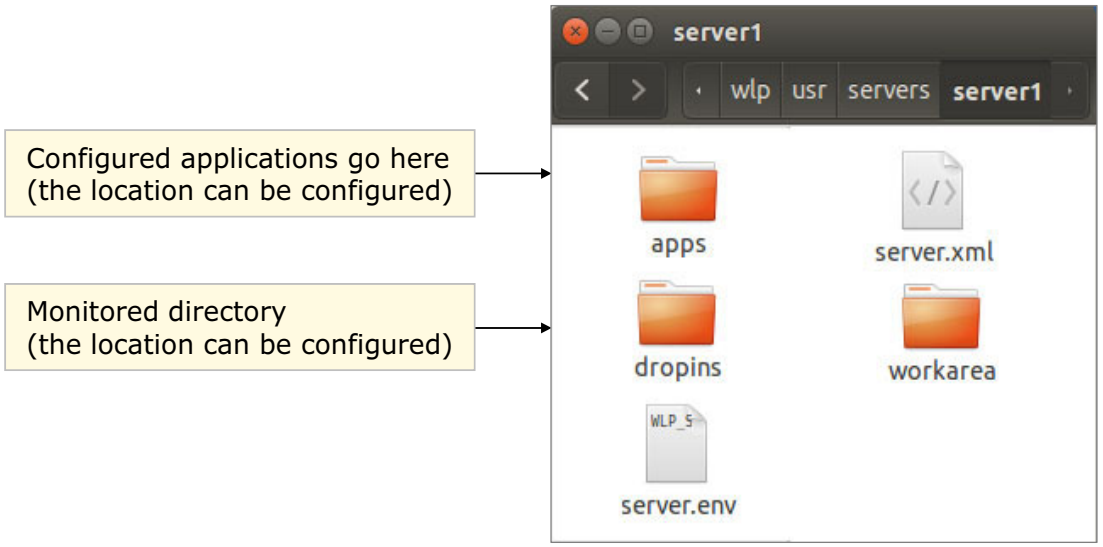






*Figure 1-17. Application deployment*

Liberty profile servers support two application deployment models:

1. Deploy an application by dropping it into the `dropins` directory
2. Deploy an application by adding it to the server configuration `server.xml` file

The `dropins` directory can be used for applications that do not require extra configuration, such as security role mapping.

If you put your applications in the `dropins` directory, you must not include an entry for the application in the server configuration. Otherwise, the server tries to load the application twice and an error might occur.

This `dropins` directory is created by default and is automatically monitored. If you drop an application into this directory, the application is automatically deployed on the server. Similarly, if the application is deleted from the directory, the application is automatically removed from the server. The `dropins` directory can be used for applications that do not require more configuration, such as security role mapping. If you put your applications in the `dropins` directory, you must not include an entry for the application in the server configuration. Otherwise, the server tries to load the application twice and an error might occur.

For applications that are not in the `dropins` directory, you specify the location by using an application entry in the server configuration. The location can be on the file system or at a URL.

Your application can be packaged as an archive file, a directory, or as a loose application where files are in multiple locations.

Deploy an application by adding it to the server configuration file. Configure the `application` element or the `webApplication` element in the `server.xml` configuration file.

An example of deploying an application by adding it to the server configuration is shown.


**Example**

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

IBM Training

## Flexible deployment: server package

- Package runtime, server configuration, and application in an archive

```
> server package myServer --include=minify
```

- Easy self-contained deployment of application, server configuration, and runtime components
  - Simple to deploy and run in any target environment

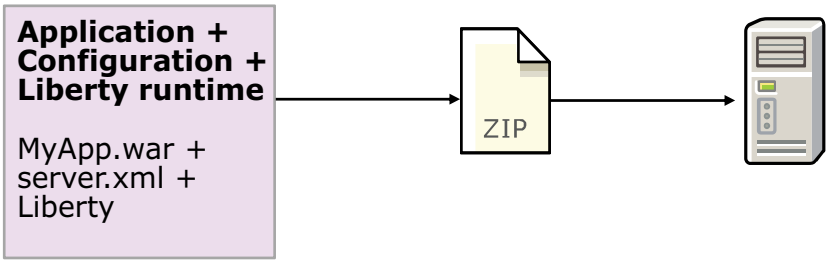






*Figure 1-18. Flexible deployment: server package*

From the command line, it is possible to create a package file that contains the Liberty runtime, the files in the shared resources directory, a specific server, and the applications that are embedded in the server. This package can be used to deploy to hosts in a Liberty collective, distribute it to colleagues, or even embed it in a product distribution. The server installation that you want to package cannot already be joined to a collective. The server to be packaged should not be running.

You can easily package a server installation in a compressed file. You can store this package, distribute it to colleagues, and then use it to deploy the installation to a different location or to another machine.

If you use the `--include=minify` option, the server 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.

Deployment of Liberty applications is easy to integrate into any delivery pipeline that can take the self-contained application and deliver to the target environment.

IBM Training

## Liberty Admin Center (1 of 3)

- Admin Center offers the ability to
  - Start and stop servers
  - View details about Liberty profile servers and applications
  - View bookmarked information
  - Add tools
  - Monitor server resources
  - Deploy server packages on hosts within a collective
- Advantages of the Admin Center design include:
  - A user interface with a mobile look-and-feel for web browsers on a cell phone or a tablet
  - Support for multiple lightweight, task-oriented tools
  - A toolbox that you can customize by selecting tools from a catalog, specifying bookmarks, or specifying user preferences




*Figure 1-19. Liberty Admin Center (1 of 3)*

The Liberty Administrative Center (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.

You can use WebSphere Liberty Administrative Center ("Admin Center") to administer servers, applications, clusters, and other resources from a web browser.

To be able to connect to the Admin Center of a Liberty server, the Admin Center feature first needs to be enabled and configured. The webProfile installation does not include the Admin Center feature by default. The adminCenter-1.0 feature enables the Liberty Admin Center, a web-based graphical interface for deploying, monitoring, and managing Liberty servers in stand-alone and collective environments.

# Liberty Admin Center (2 of 3)

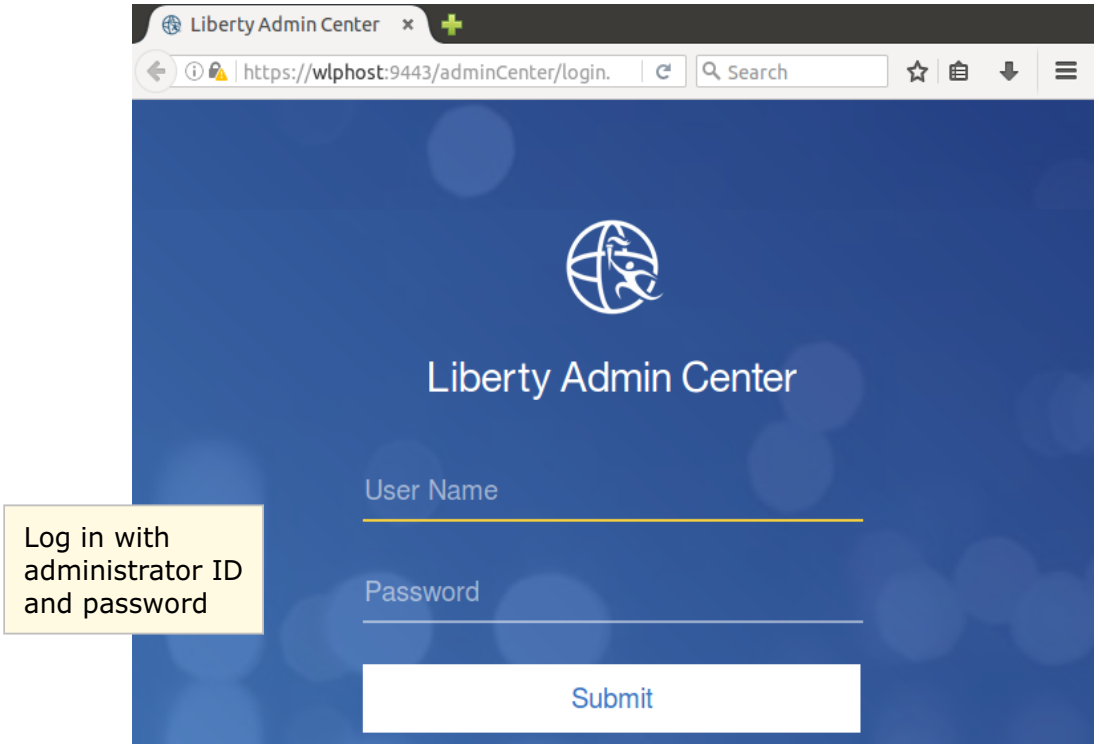






*Figure 1-20. Liberty Admin Center (2 of 3)*

Log in to the Admin Center by specifying an authorized user name and password.

To start the Admin Center, enter: `http://<host>:<port>/adminCenter/`

# Liberty Admin Center (3 of 3)

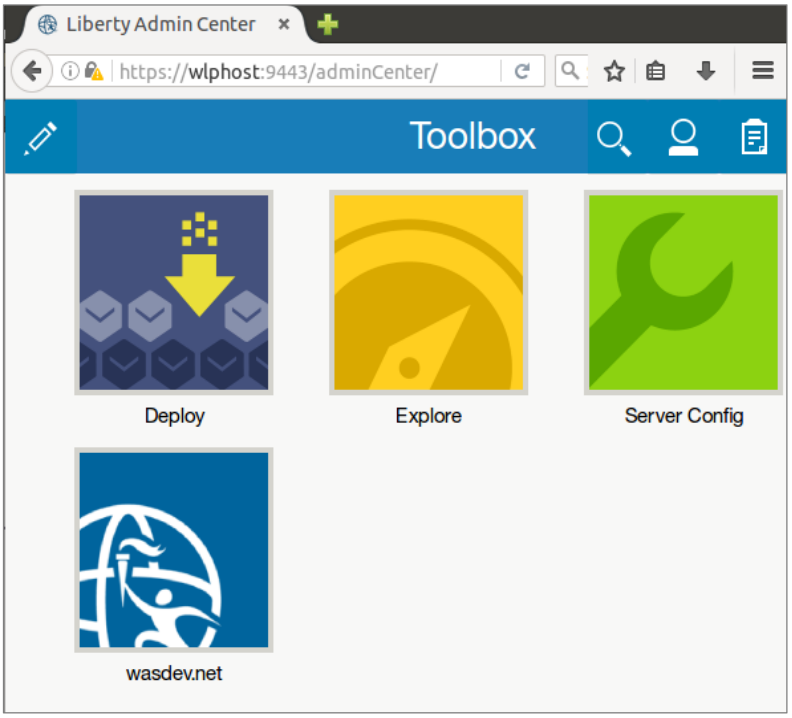






*Figure 1-21. Liberty Admin Center (3 of 3)*

The Admin Center Toolbox is displayed by default when you sign on.

The **Deploy** tool is how you deploy a server package that includes a Liberty profile and a server.

The **Explore** tool provides a summary of all applications, clusters, servers, hosts, or runtimes.

- You can view details of each of the resources from the Explore tool.
- You can start, stop, or restart a resource.

You can **Search** for resources.

The **Server Config** tool is used to view or edit server configuration files in the Liberty topology.

You can view the Liberty Repository from the **wasdev.net** icon.

The **Edit** icon is used to add or remove tools and bookmarks.

## Liberty administrative security

- One administrator role
- One user registry for applications and administrators
- Simple configuration for a single administrator user

```
<quickStartSecurity userName="admin"
                    userPassword="{xor}Lz4sLCgwLTs"/>

<keystore id="DefaultKeyStore"
                password="{xor}DFoKyp="/>
```

- Also, an easy configuration for multiple users

```
        <administrator-role>
            <user>bob</user>
            <group>administratorsGroup</group>
        </administrator-role>
```




*Figure 1-22. Liberty administrative security*

You can use the `<quickStartSecurity>` element to quickly enable a simple (one user) security setup for Liberty.

The administrator role has authorization access to the graphical admin center and to run scripts.

More users can be added to the administrators group.

# 1.2. Commonly used server commands

IBM Training

# Commonly used server commands




*Figure 1-23. Commonly used server commands*

IBM Training

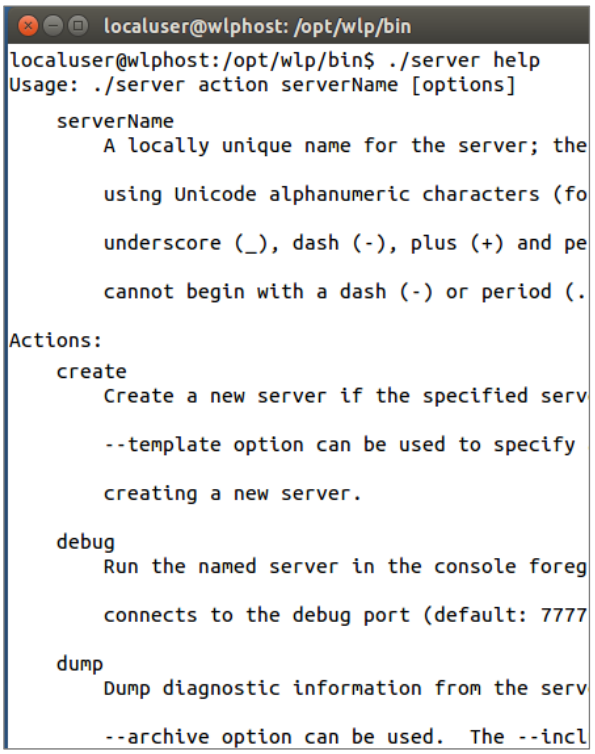
## Topics

- Liberty overview
- Commonly used server commands




*Figure 1-24. Topics*

# Terminal commands




- From `<wlp_root>/bin`, you can type server commands
- Type `server help` for a list
  - `create`
  - `debug`
  - `dump`
  - `javadump`
  - `list`
  - `package`
  - `run`
  - `start`
  - `status`
  - `stop`
  - `version`




*Figure 1-25. Terminal commands*

The slide shows the commands that can be typed from the terminal window.

The `<wlp_root>` directory is the directory of the Liberty runtime installation.

IBM Training

## Command format and examples

- The format is: `server <action> [serverName] [options]`
- To create a server: `server create server1`

```
$ server create server1
Server server1 created
```

- To start a server: `server start server1`

```
$ server start server1
Starting server server1.
Server server1 started with process ID 17706.
```




*Figure 1-26. Command format and examples*

The server command supports starting, stopping, creating, packaging, and dumping a Liberty server.

The command syntax is as follows:

`server` *`action`* `serverName [options]`

Where, the value of action represents the operation that you can perform on a Liberty server.

You can create a server from the command line. Open a command line, then go to the `wlp/bin` directory. Run the `server create` command to create a server. If you do not specify a server name, defaultServer is used. Where *server_name* is the name you want to give your server. If the server is created successfully, you receive message: `Server server_name created`.

A directory with the name of the new server is created under the `${wlp.user.dir}/servers` directory, containing the configuration of the new server. The HTTP port numbers for the new server are assigned to default values and are shown in the generated `server.xml` file to make it easy to edit them. You can also set these values by using variables in a `bootstrap.properties` file in the same directory.

The `server start` command is a command that starts the server as a background process.

IBM Training

## More server actions

- You can list servers and check their status

```
$ server list
The following servers are defined relative to the
user directory /opt/wlp/usr.

adminCenterController
server1
static_1
static_2

$server status server1
Server server1 is running with process ID 17706.
```

- You can check the version of Liberty

```
$ server version
WebSphere Application Server 16.0.0.2
(1.0.13.cl160220160526-2258) on IBM J9 VM, version
pxa6480sr3-20160428_01 (SR3) (en_US)
```




*Figure 1-27. More server actions*

Other server actions include listing the servers and checking their status. You can also check the version of Liberty that is installed.

## Start a server by using the run action

- Type: `server run [serverName]`
- The server starts in the foreground terminal and messages are streamed to the console
  - Useful for verifying that the server starts correctly
  - Useful for finding URLs of web applications

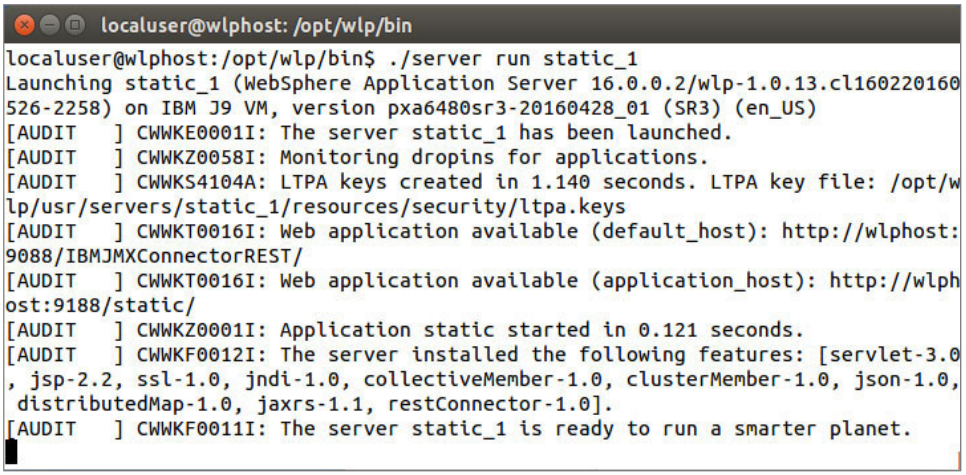


*Figure 1-28. Start a server by using the run action*

You can start a server with the `server run [serverName]` command.

The server starts in the foreground console and `messages.log` file messages are streamed to the console.

- Useful for verifying that the server starts correctly
- Useful for finding URLs of web applications

IBM Training

# Unit summary

- Describe Liberty
- Describe the composable architecture of Liberty
- Describe the Liberty configuration file
- Describe the continuous delivery of new Liberty function
- Describe the flexible deployment of applications
- Describe the dynamic nature of the configuration
- Describe the manual administration of Liberty servers and features




*Figure 1-29. Unit summary*

## Review questions

1. True or False: The Liberty profile is freely available.

2. Which server command do you use to stream the output to the terminal?
   A. `server run [serverName]`
   B. `server start [serverName]`
   C. `server list [serverName]`
   D. `server status [serverName]`




*Figure 1-30. Review questions*

Write your answers here:

1.

2.

## Review answers

1. True or False: The Liberty profile is freely available.
   The answer is True.

2. Which server command do you use to stream the output to the terminal?
   A. `server run [serverName]`
   B. `server start [serverName]`
   C. `server list [serverName]`
   D. `server status [serverName]`
   The answer is A.




*Figure 1-31. Review answers*

# Unit 2. Multi-server management

## Estimated time

01:30

## Overview

In this unit, you learn the various options for managing Liberty servers. You use the Liberty collective management model to manage servers.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

IBM Training

## Unit objectives

- Describe the Liberty collective architecture
- Describe how to create and configure a Liberty collective
- Describe the use of clusters in a collective
- Describe how to configure and manage a Liberty server cluster
- Describe how the Admin Center works with different topologies
- Describe the ways that Liberty servers can be managed in cloud environments




*Figure 2-1. Unit objectives*

# 2.1. Liberty collective management model

IBM Training

# Liberty collective management model




*Figure 2-2. Liberty collective management model*

IBM Training

## Topics

- Liberty collective management model
- Creating a Liberty collective
- Setting up a Liberty cluster
- Liberty Admin Center
- Alternative Liberty management models in cloud environments




*Figure 2-3. Topics*

IBM Training

## How do you manage all these servers?

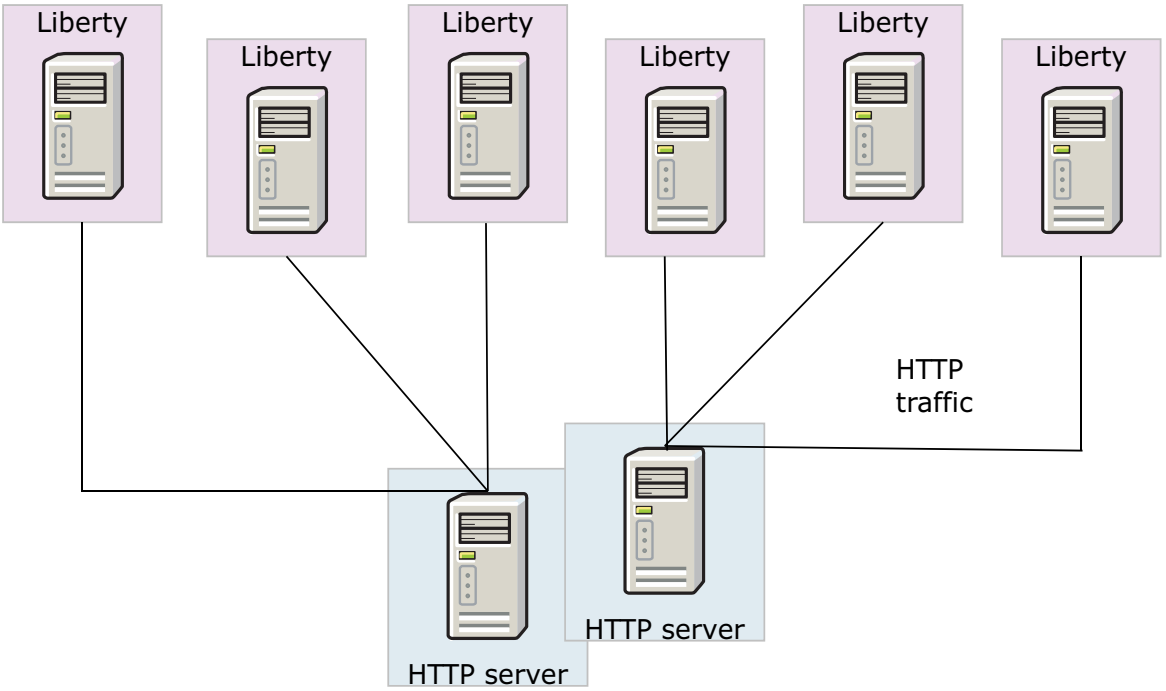






*Figure 2-4. How do you manage all these servers?*

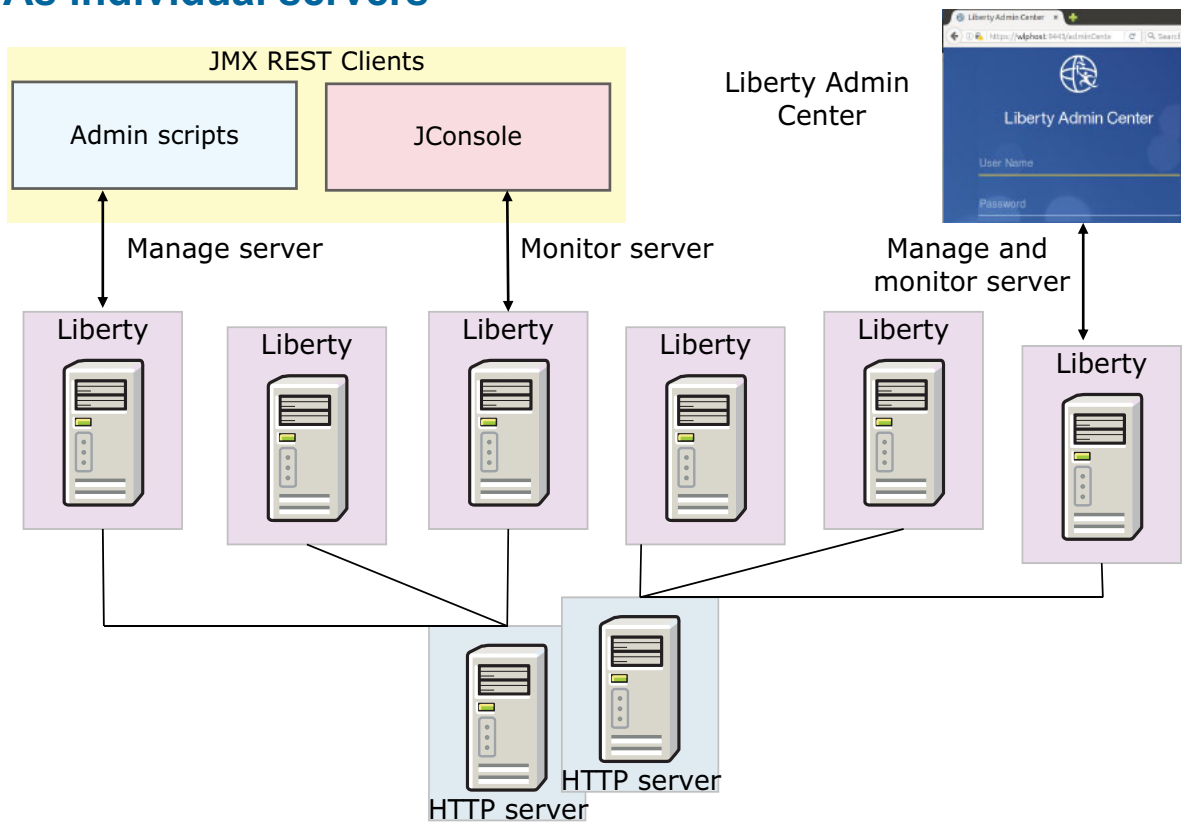
One of the benefits of Liberty is that you have a wide choice of options when configuring a deployment environment. You can deploy Liberty as a stand-alone individual server or as part of a collective, which is used to manage multiple servers from a single management domain. You can also deploy Liberty as part of a traditional WebSphere Application Server cell. Liberty, being small, easy to run and use, and flexible, it lends well to running in the cloud. You can deploy Liberty in a platform as a service (PaaS), various on-premises cloud offerings, and in containers. Liberty is being designed to run in any environment that you want to run it in.

Individual developers typically work in single-server environments, but at the same time, Liberty is also suited for production. Using multiple Liberty servers can provide the availability and scalability for running critical applications.

In a multi-server environment, the challenge becomes how to manage the servers effectively.

IBM Training

## As individual servers




Multi-server management



*Figure 2-5. As individual servers*

Typical administration actions in a Liberty environment include creating, starting, and stopping a server, querying the status of a server, packaging a server, and complete troubleshooting actions, such as creating a memory dump for diagnosis.

Liberty servers are administered by using line commands, scripts, the Admin Center Web UI, APIs, or from the WebSphere developer tools. Liberty Management APIs include Java Management Extensions (JMX) MBeans and Representational State Transfer (REST) application programming interfaces (APIs).

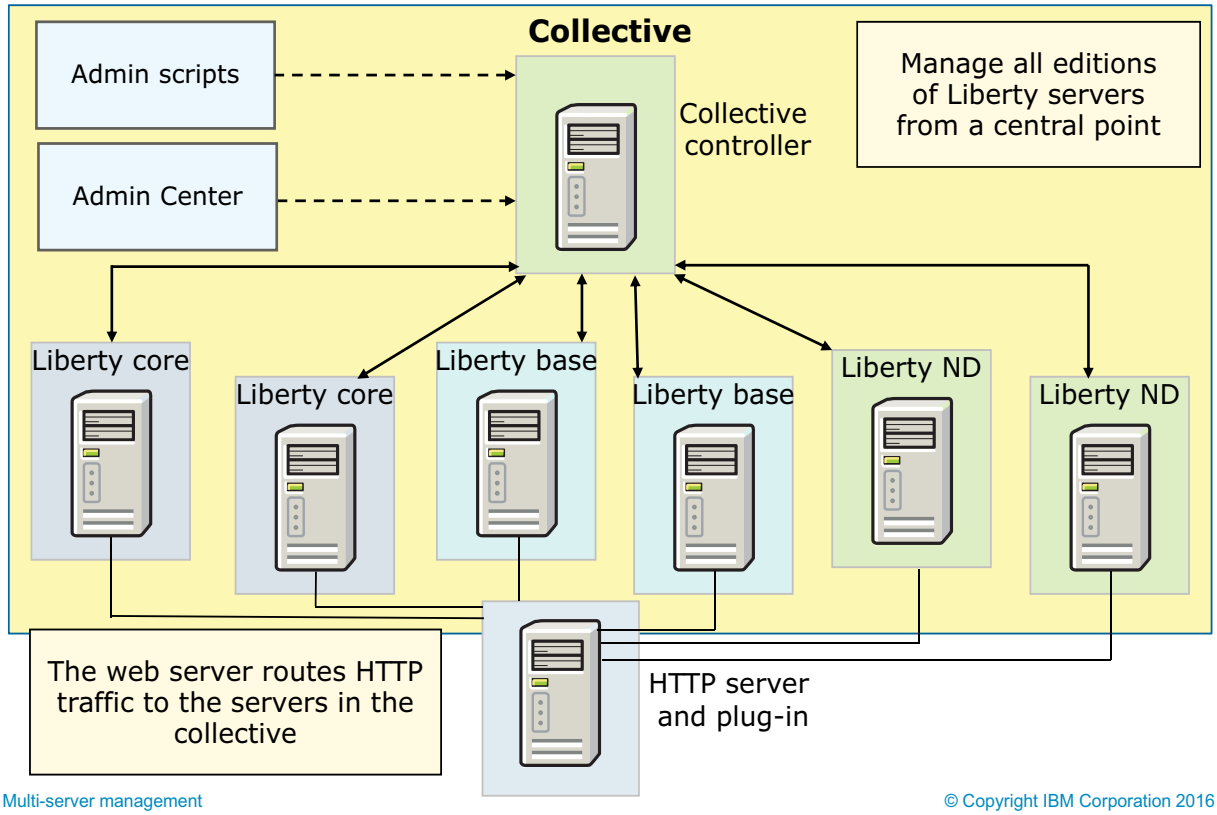
The operation targets for the JMX (Java Management Extensions) REST clients and the Liberty Admin Center console are individual servers.

Java Management Extensions is a Java technology API for managing and monitoring servers and applications.

The WebSphere Liberty Administrative Center (Admin Center) feature is available for Liberty, which provides a web-based graphical interface for Liberty servers and resource management. It is designed around a toolbox model, so you can select tools in a customized Admin Center instance.

IBM Training

## With a centralized management model




Multi-server management



*Figure 2-6. With a centralized management model*

Liberty servers can now be administered as a part of a common management domain, called a Liberty collective. This structure is added to the administration options for Liberty for operational efficiency and convenience and to introduce high availability features.

A collective comprises at least one Liberty server that is configured as a collective controller and possibly one or more Liberty servers that are configured as collective members.

The Liberty Collective is a multi-server administrative domain that is designed exclusively for the WebSphere Application Server Liberty Profile. Liberty Collectives were introduced with WebSphere Application Server V8.5.5.

A Liberty Collective extends the Liberty management model that is provided in WebSphere Application Server V8.5 by enabling servers to be grouped into a “collective”.

The collective controller provides a central point for you to manage servers.

Each server has its own configuration and can easily join or leave the collective.

The Liberty controller requires a Liberty Network Deployment license.

IBM Training

## What is a collective?

- A collective is the set of Liberty servers in a single administrative domain
- A collective consists of at least one collective controller, which is any Liberty profile server with the `collectiveController-1.0` feature enabled
- Optionally, a collective can have several collective members, which are servers with the `collectiveMember-1.0` feature enabled
- You can configure a collective to have many collective controllers, called a replica set




*Figure 2-7. What is a collective?*

A collective is the set of Liberty servers in a single administrative domain.

A collective consists of at least one collective controller, which is any Liberty profile server with the `collectiveController-1.0` feature enabled.

The collective controller provides for a centralized administrative control point to perform operations such as routing, file transfer, and cluster management.

Membership in a Liberty collective is optional. For a member to be part of a collective, it must join a collective controller. A member can join only one collective. The collective can have more than one controller for failover and workload balancing reasons, but the member only communicates with one controller at a time. The communication between the member and the controller is done over the IBM JMX Rest Controller with MBean operations.

A Liberty server that is configured as the collective controller can optionally provide full lifecycle management to all members in the collective. Which includes product installation and maintenance, and operational access to all servers in the collective, without requiring an agent. The collective controller includes operations to start and stop servers, invoke administrative operations, and perform file transfer in support of configuration changes and application installation. All Liberty servers can be members of a collective, but only Network Deployment or WebSphere Application Server for z/OS provide the support that is needed to create a collective controller.

A collective can have several collective members, and can configure a collective to have many collective controllers, called a replica. You can have only one replica set per collective, and all controllers must be part of the replica set.

IBM Training

## What is a replica set?

- A replica set is a set of collective controllers that are configured to work together.
  - Each replica contains all the repository updates that the other replicas within the set have processed.
- A replica set provides highly available management capabilities for a Liberty administrative domain.
- A replica set must have at least three replicas, preferably on different hosts, for high availability.
  - When the replicas are on the same host, you must assign unique port numbers for each of the replicas.




*Figure 2-8. What is a replica set?*

A set of collective controllers is called a replica set. There can be only one replica set per collective and all controllers must be part of it. When there is more than one collective controller, each collective controller replicates its data to the other collective controllers in the replica set to allow for high availability and data protection. The replica set is logically present even when only one controller is in use. Liberty servers in a collective can be clustered to provide scalability and availability of applications. The cluster can be treated as a single object in the collective, simplifying the operational management of the servers in the cluster. The members of the cluster can be configured individually, or can share a configuration. A single collective can have multiple clusters, but a server can only be part of one cluster at a time.

IBM Training

# Example topology diagram for a collective

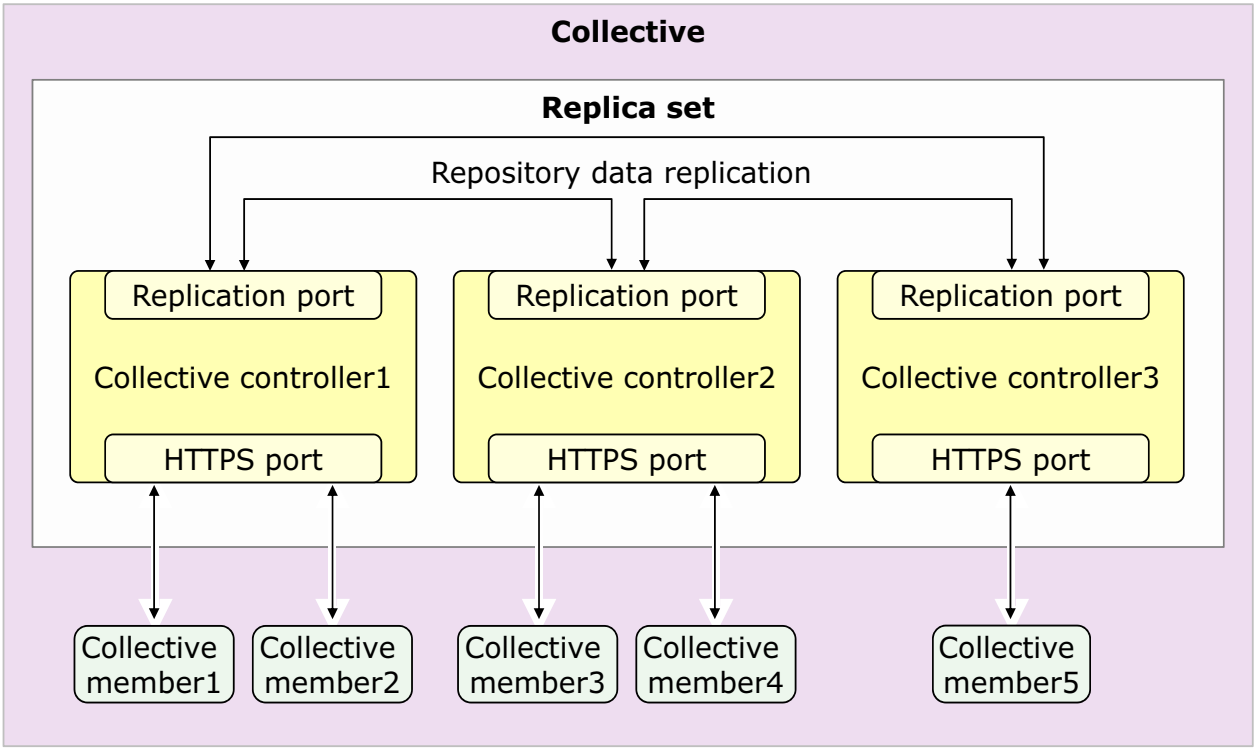






*Figure 2-9. Example topology diagram for a collective*

When there is more than one collective controller, each collective controller replicates its data to the other collective controllers in the replica set to allow for high availability and data protection. When multiple replicas are configured in a set, include at least three replicas in the set. The controllers in the replica set communicate with each other by using a collaboration scheme to ensure that data is replicated across the set of controllers. Therefore, any controller in the set can receive an operation to store data. Each controller has a dedicated port for use by the replication protocol. Communication between the controllers in the replica set is always authenticated and protected with SSL.

A collective member can be configured with multiple collective controller endpoints. A collective member communicates only with one collective controller at a time; however, a configuration with more than one collective controller endpoint provides failover and workload balancing. Member-to-controller communication is always in the form of MBean operations that are completed over the IBM JMX Rest Connector. Communication between controllers and members is always authenticated and protected with SSL.

IBM Training

## Liberty Profile systems management is

- **Automated:** Less input is required to establish a secure configuration.
- **Fast:** No profile management tool. A simpler-to-use command-line tool set is used instead.
- **Easy:** Easy to script as functions are consistently command-line enabled.
- **Self-contained:** Servers can be archived, moved to new location, and joined to a collective quickly.
- **Agent-less:** No node agents means less cost on system resources to enable centralized management.
- **Decentralized:** The server owns its own configuration.




*Figure 2-10. Liberty Profile systems management is*

Liberty Profile systems management is designed for speed, ease of use, and flexibility.

The Liberty collective was built on five core principles:

1. Administration should be exposed through a standards-based API.
2. All administration for Liberty is provided through MBeans, which enables a common set of tools to complete administrative actions.
3. The entities within a collective should be loosely coupled to the collective. Therefore, all configuration that is related to the collective is isolated and self-contained, which facilitates easily moving servers in and out of the collective.
4. The administrative server, called a collective controller, acts as a distributed cache. Application servers within the collective publish information to the controller about themselves, such as which applications are installed, their operational state, and the available MBeans.

   Each member in the Liberty collective owns its own configuration. There is no central, master repository of configuration as there is in the full profile cell.
5. The administrative server is highly scalable and highly available through a replica model, allowing multiple instances of controllers to share data and complete the same operations. The

model is agentless, which means a separate agent process is not required on each host system within the collective.

# 2.2. Creating a Liberty collective


*Figure 2-11. Creating a Liberty collective*

IBM Training

## Topics

- Liberty collective management model
- Creating a Liberty collective
- Setting up a Liberty cluster
- Liberty Admin Center
- Alternative Liberty management models in cloud environments




*Figure 2-12. Topics*

IBM Training

## Steps for creating a simple collective

1. Create the controller server
2. Create the collective configuration
3. Update the controller configuration
4. Start the controller
5. Create the first member
6. Join the member
7. Update member configuration
8. Start the member server

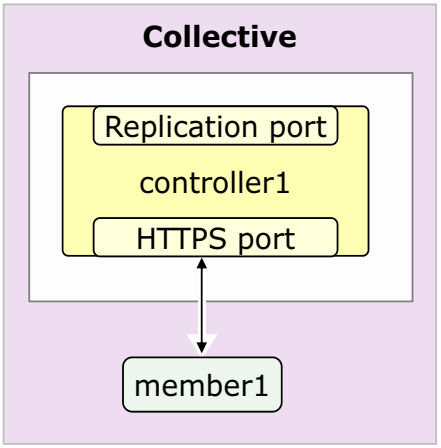






*Figure 2-13. Steps for creating a simple collective*

This slide lists the necessary steps for creating a basic collective.

Most of these steps are completed by using the server and collective commands, which are explained throughout the presentation. Some steps require that you manually edit server configuration files.

## Create the controller server

- You can configure any Liberty server to function as a controller
- Use the `server create` command to create the server that acts as the collective controller
- For example:
  - `<wlp_root>/bin/server create controller1`

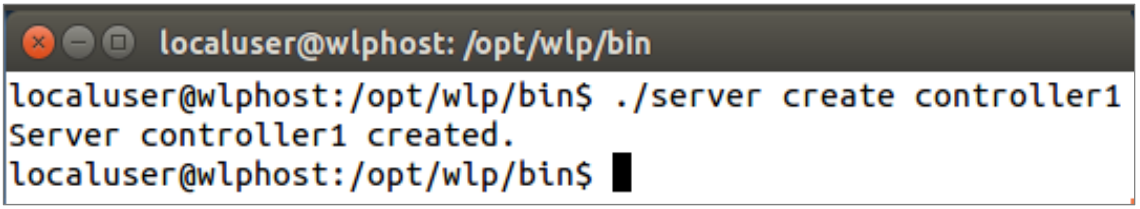






*Figure 2-14. Create the controller server*

The collective controller acts as a command and control mechanism for the administrative functions of the collective. The collective controller also serves as a storage and collaboration mechanism for the collective and cluster members. The collective controller is a standard Liberty server with the collective controller feature enabled. You can configure any Liberty server to function as a controller.

During this step, you create a server by using the defaultServer template. There is nothing special about this server until more configuration is created to make it a functioning collective controller.

You can give any unique name for the server that acts as the collective controller. In the example, the name that is given is *controller1*.

To create a collective controller, the following procedure should be used:

1. Create a Liberty server.
2. Configure the server as a collective controller.
3. Configure administrative security.
4. Enable the Admin Center feature for the collective controller.

## Create the collective configuration

- The `collective create` command establishes the administrative domain security configuration
  - For example: *<wlp_root>*`/bin/collective create controller1 --keystorePassword=passw0rd --createConfigFile --hostName=wlphost`

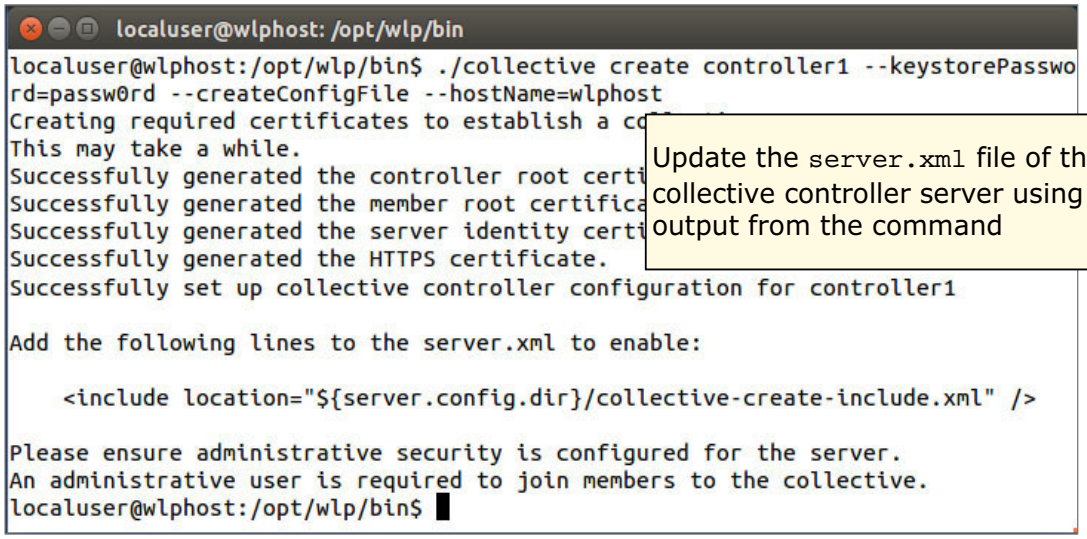






*Figure 2-15. Create the collective configuration*

The collective create command establishes the administrative domain security configuration.

In this step, two actions occur. First, required security certificates are generated for the collective. These certificates include:

- Root certificates for the controller and future members
- A server identity certificate
- An HTTPS certificate

Second, if the optional `-createConfigFile` parameter is omitted, configuration elements are generated as output to the console that must be manually copied to the `server.xml` file for the controller server. When the `-createConfigFile` is specified as a parameter on the create command, a file that is named `collective-create-include.xml` is generated. The element `<include location=${server.config.dir}/collective-create-include.xml>` must be added to the `server.xml` for the controller manually.

IBM Training

## collective-create-include.xml file (1 of 2)

```
 <?xml version="1.0" encoding="UTF-8" ?>
<server description="This file was generated by the 'collective
  create' command on 2016-10-06 11:29:47 EDT.">
    <featureManager>
        <feature>collectiveController-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" />

    <!-- TODO: Set the security configuration for Administrative
  access -->
    <quickStartSecurity userName="" userPassword="" />
```

You must manually type the values for the `userName` and `userPassword`. These values are generated as empty strings

Values for the `adminUser` and `adminPassword` variables can be set with variables defined in a `bootstrap.properties` file




*Figure 2-16. collective-create-include.xml file (1 of 2)*

The next two slides show the output that the collective create command generates into the `collective-create-include.xml` file when the `-createConfigFile` is specified as a parameter on the collective create command. Notice that some elements are flagged with “TODO” comments. On this slide, the `quickStartSecurity` element is shown with variable values that are assigned to user name and user password. In the generated file, these values are empty strings.

You can set the userName and userPassword values to variables that are defined in a separate `bootstrap.properties` file. You must manually create the `bootstrap.properties` file in the same directory as the `server.xml` of the collective controller.

You can write the output of this collective command to a file, instead of to a console screen by specifying an optional `-createConfigFile` parameter. You can then include the outputted file in the collective configuration by adding an include statement to the `server.xml` file.

**Example**

Set the userName and userPassword to variables that are defined in `bootstrap.properties`:

```
<quickStartSecurity userName="${adminUser}" userPassword="${adminPassword}" />
```

IBM Training

## collective-create-include.xml file (2 of 2)

```
<!-- clientAuthenticationSupported set to enable bidirectional trust -
  ->
 <ssl id="defaultSSLConfig"
      keyStoreRef="defaultKeyStore"
      trustStoreRef="defaultTrustStore"
      clientAuthenticationSupported="true" />

 <!-- inbound (HTTPS) keystore -->
 <keyStore id="defaultKeyStore" password="{xor}Lyg7"
           location="${server.config.dir}/resources/security/key.jks"
  />

 <!-- inbound (HTTPS) truststore -->
 <keyStore id="defaultTrustStore" password="{xor}Lyg7"

  location="${server.config.dir}/resources/security/trust.jks" />
```

The files named `key.jks` and `trust.jks` are generated to the `/resources/security` folder of the collective controller




*Figure 2-17. collective-create-include.xml file (2 of 2)*

The default keystore and truststore passwords that are shown on this slide are encoded from the value of keystorePassword that was specified with the collective create command.

The generated keystore and truststore files that are written in the `/resources/security` subdirectory of the collective controller.

IBM Training

## Update the controller configuration

- Typically, you can update the `server.xml` file of the controller by opening it with a text editor and copy/paste the output from:
  - The command terminal window
  - A specified output file
- Enable the controller to use the Admin Center:
  - Add `<feature>adminCenter-1.0</feature>` to the `featureManager` element
- Remember to add the location element of the generated collective config file to the `server.xml` file

```
<include location="${server.config.dir}/collective-create-
  include.xml" />
```




*Figure 2-18. Update the controller configuration*

You can keep all your configuration in a single `server.xml` file, or you can use include elements to consolidate configurations from separate files to create the structure that is most useful to you.

Typically, you can update the `server.xml` file of the controller by opening it with a text editor and copy and paste the output from the command terminal window.

It can be easier to maintain a complex configuration by splitting it across a set of files. For example:

You might want to include a file that contains variables that are specific to the local host, so that your main configuration can be used on multiple hosts.

You might want to keep all of the configuration for a particular application in a separate file that can be versioned with the application itself.

You can use the `include` element to include configuration information from an external xml file in the `server.xml` file.

You can configure the Admin Center feature for any Liberty profile server. In a collective, it is only necessary to configure the controller server with the Admin Center feature.

To enable the Admin Center, add the adminCenter feature to the featureManager element.

Remember to add the location element of the generated collective config file in the `server.xml` file.

The `collective-create-include.xml` file is required to enable the server to function as a collective controller.

# The server.xml file for the controller

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>adminCenter-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"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

   <include location="${server.config.dir}/collective-create-
  include.xml" />
</server>
```




*Figure 2-19. The server.xml file for the controller*

The page shows an example of the `server.xml` for the controller.

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 contains the statement:

```
<feature>collectiveController-1.0</feature>
```

IBM Training

## Sample bootstrap.properties file

- The `bootstrap.properties` file can be used to specify variables
  - Administrator user and password
  - Host name of the computer
  - Port definitions

The `server.xml` file

```
<server description="new server">
    <featureManager>
        <feature>collectiveController-1.0
        </feature>
    </featureManager>

    <httpdEndpoint id="defaultHttpEndpoint"
        host="*"
        httpPort="${httpAdminPort} "
        httpsPort="${httpsAdminPort}" />
```

The `bootstrap.properties` file

```
httpAdminPort=9080
httpsAdminPort=9443
```




*Figure 2-20. Sample bootstrap.properties file*

Bootstrap properties initialize the runtime environment for a particular server. Generally, they are attributes that affect the configuration and initialization of the runtime core.

Bootstrap properties are set in a text file named `bootstrap.properties`. This file is not required, so it does not exist unless you create it. You must create this file in the server directory, which also contains the configuration root file `server.xml`. By default, the server directory is `usr/servers/`*`server_name`*.

You can create a `bootstrap.properties` file by using the editor in WebSphere Application Server Developer Tools for Eclipse. From the **Servers** view, right-click the server you want to configure, then click **New > Server Environment File**, then `bootstrap.properties`, and the file is created from a template and opened in an editor. Along with the `server.xml` and `server.env` files, the `bootstrap.properties` file appears in the **Servers** view under the server that it is associated with and can be edited by double-clicking it.

You can edit the `bootstrap.properties` file by using a text editor or the editor in the WebSphere Application Server Developer Tools for Eclipse. If you update the `bootstrap.properties` file, you must restart the server for the changes to take effect.

The `bootstrap.properties` file contains two types of properties:

- A small, predefined set of initialization properties.
- Any custom properties that you choose to define.

You can then use these custom properties as variables in other configuration files such as `server.xml` and included files.

## Files and folders on the controller directory

- The files are now on the controller directory
  - `server.xml`
  - `collective-create-include.xml`
  - `bootstrap.properties`

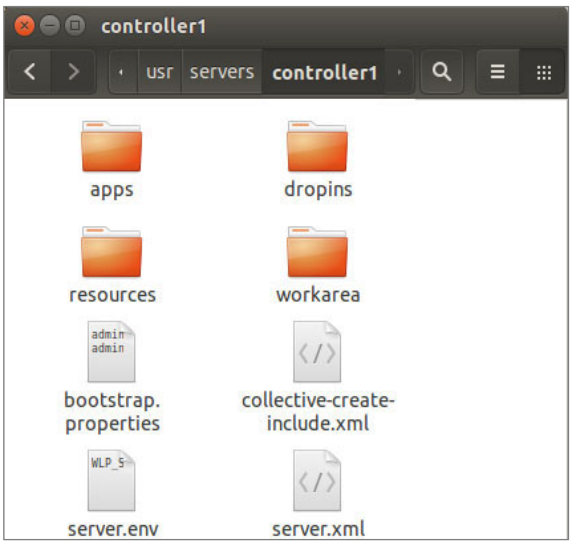






*Figure 2-21. Files and folders on the controller directory*

Runtime services provide their configuration defaults so that the configuration you need to specify is kept to a minimum. At server start (or when the user configuration files are changed), the kernel parses your configuration and applies it over the system defaults. The set of configuration properties that belong to each service is injected into the service each time the configuration is updated.

The files that can be in the `<wlp_root>/usr/servers/<controller>` directory:

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

The `server.xml` is created when the server is created. The `collective-create-include.xml` is generated by the collective create command. The `bootstrap.properties` file is manually created.

A Liberty server can be customized by using a few simple files:

- `server.xml:` The primary configuration file for the Liberty server. This file is the one non-optional configuration file. It has a simple XML format that is suitable for text editors.

- `bootstrap.properties`: An optional text file that is used to customize the kernel bootstrap process or to specify more variables for use in `server.xml`.
- `jvm.options`: An optional file that is used to specify JVM options for the server. If this file is present, it supersedes the `jvm.options` file in the `/etc` directory (only one file is used).
- `server.env`: An optional file that is used to customize environment variables that are used to start the server. If both this file and the `/etc/server.env` file are present, the contents of both are merged together with values specific to the server superseding values specified for the installation. The optional files are not created by default, whereas `server.xml` is always created.

## Start the controller

- The collective controller is started just like any other Liberty server
- Use the `server start` or `server run` commands
- For example:
  - *<wlp_root>*`/bin/server start controller1`
- Verify that the server starts correctly by viewing the `messages.log` file
  - *<wlp_root>*`/usr/servers/controller1/logs/messages.log`
- Look for the following messages:

```
CWWKX9000I: The CollectiveRepository MBean is available.
CWWKX9003I: CollectiveRegistration MBean is available.
```




*Figure 2-22. Start the controller*

The collective controller is started just like any other Liberty server by using the `server start` command.

You can also start a Liberty profile server by using the `server run` command. The result of this command is to run the server in the foreground so that all messages that are streamed to the `console.log` file are also written to the terminal window. Starting a server with the run command is useful for seeing the URLs for applications that run on the server.

Verify that the server starts correctly by viewing the messages.log file. Look for the log messages that display the messages:

`CWWKX9000I: The CollectiveRepository MBean is available`

`CWWKX9003I: CollectiveRegistration MBean is available`

IBM Training

## Create a member and join it to the collective

- You can join any Liberty server to a collective
  - If the Liberty server runs on the same host as the controller, you must manually edit its `server.xml` file to make the HTTP and HTTPS ports unique

```
<httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9081"
                  httpsPort="9444" />
```

- For example, if you have a Liberty server that is named `member1`, join the server to the collective as follows:
  - `/bin/collective join member1 --host=wlphost --port=9443 --user=admin --password=passw0rd --keystorePassword=passw0rd`
- The host name of the controller and its HTTPS port must be specified along with user ID and password for `quickStartSecurity`
  - The controller must be running
- The output from this command is the configuration that you must use to update the `server.xml` file of the member server

Multi-server management



*Figure 2-23. Create a member and join it to the collective*

You can join any Liberty server to a collective by using the `collective join` command. This slide shows an example of the `collective join` command.

The output from this command is the configuration that you must use to update the `server.xml` file of the member server.

Recall that the `server create` command uses a default template in which the HTTP port is 9080 and the HTTPS port is 9443. If you create multiple servers on the same host, you must manually update these ports in the `server.xml` file so that there are no port conflicts with other servers.

The `collective join` command requires an SSL session between the client (command script) and the target server (controller), so the SSL handshake requires you to accept the controller's certificate.

You must accept the certificate for the join process to complete successfully.

IBM Training

# Update the collective member configuration

```
Successfully joined the collective for server member1

Add the following lines to the server.xml to enable:
    <featureManager>
        <feature>collectiveMember-1.0</fea
    </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" />

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

Output from the collective join command. These elements must be added to the `server.xml` file of collective member1

The name of the host and https port of the controller




*Figure 2-24. Update the collective member configuration*

The slides show the output from the `collective join` command that is written to the terminal.

This section of the configuration includes the `collectiveMember` feature, the host name for the collective, and remote host authentication data.

These XML elements must be added to the `server.xml` file of member1.

## Start the collective member

- The collective member is started just like any other Liberty server
- Use the `server start` or `server run` commands
- For example:
  - `<wlp_root>/bin/server start member1`
- Verify that the server starts correctly by viewing the `messages.log` file
  - `<wlp_root>/usr/servers/member1/logs/messages.log`
- Look for the following messages:

```
CWWKX8112I: The server's host information was successfully
published to the collective repository.
CWWKX8114I: The server's paths were successfully published to the
collective repository.
CWWKX8116I: The server STARTED state was successfully published
to the collective repository.
```




*Figure 2-25. Start the collective member*

The messages that are written to the messages log during server startup are the best way to validate that the server is running properly.

## Basic collective topology

- You now have a basic collective topology
  - member1 is a collective member
  - controller1 is the collective controller
- All collective members publish information about themselves to their collective controller
  - Published information is available for query directly from the controller without need of forwarding the request down to each collective member

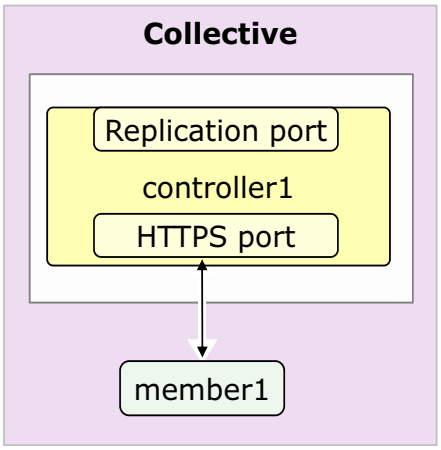






*Figure 2-26. Basic collective topology*

You now have a basic collective topology; member1 is a collective member and controller1 is a collective controller.

All collective members publish information about themselves to their collective controller. Published information is available for query directly from the controller without need of forwarding the request down to each collective member. The collective controller stores the member information in a memory cache.

# 2.3. Setting up a Liberty cluster

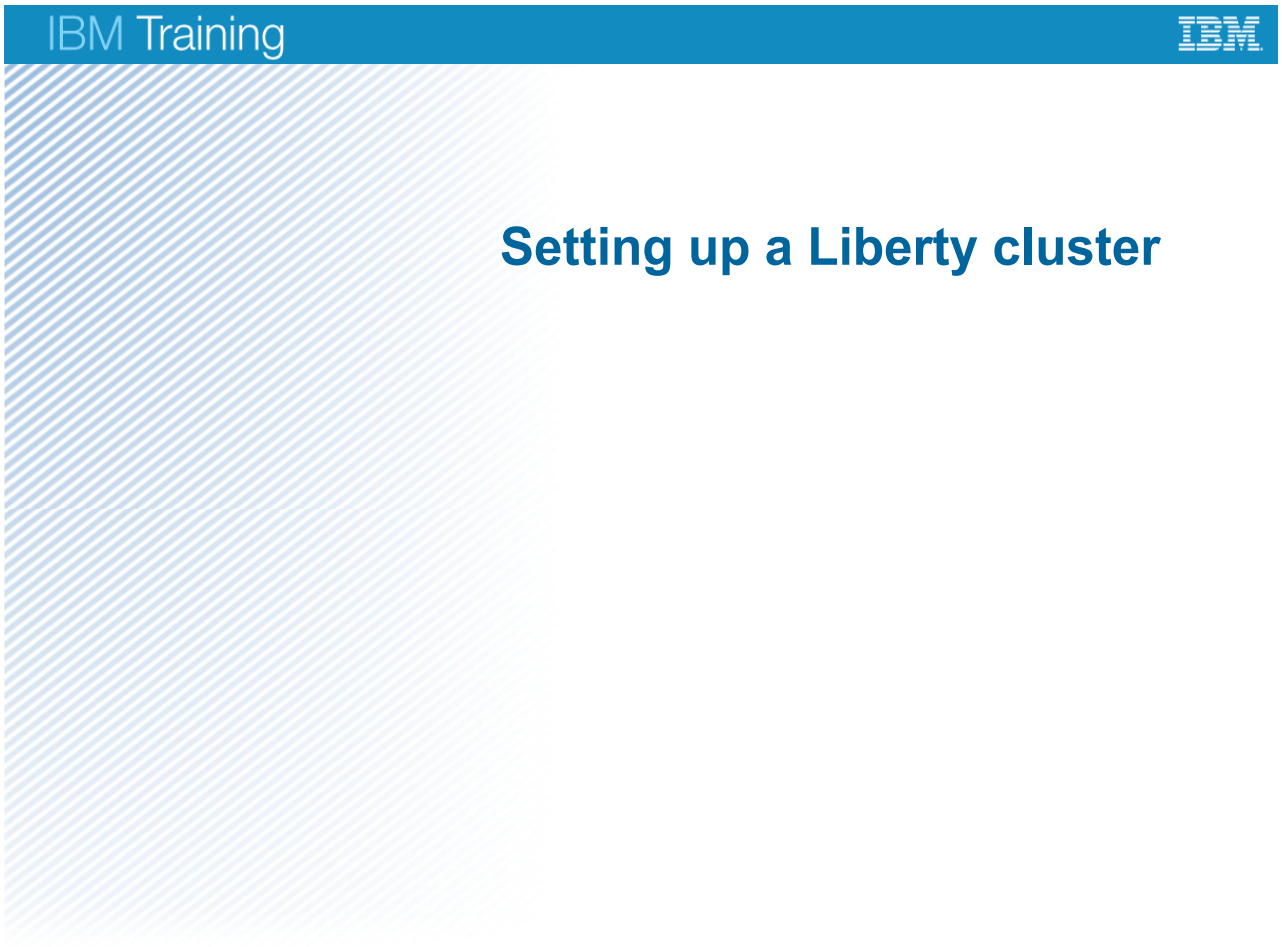



*Figure 2-27. Setting up a Liberty cluster*

IBM Training

## Topics

- Liberty collective management model
- Creating a Liberty collective
- Setting up a Liberty cluster
- Liberty Admin Center
- Alternative Liberty management models in cloud environments




*Figure 2-28. Topics*

## Liberty clusters

- A Liberty profile can be configured into a server cluster to ensure:
  - Application high availability
  - Scalability
- A Liberty cluster is a logical grouping of related servers
  - Same applications
  - Same business unit
  - Same administrators
- You must define a Liberty cluster within a Liberty collective
- All Liberty cluster members must be members of the same collective
- One collective can have multiple Liberty clusters

- The `collectiveMember-1.0` feature is available in all Liberty Profile editions
  - Therefore, servers from all editions can be members of a collective
- The Liberty cluster feature `clusterMember-1.0` is available only in the Network Deployment Liberty Profile edition




*Figure 2-29. Liberty clusters*

A Liberty profile server can be configured into a server cluster for more efficient management of servers that are hosting the same applications. The cluster provides high availability and scalability for the applications. To enable a cluster, the `clusterMember-1.0` feature needs to be available in the Liberty installation.

A Liberty server cluster consists of two or more Liberty servers within a Liberty collective. Logically grouped servers might have the same applications. The group of servers might be associated with the same business unit, or have the same administrators.

The cluster name is a string value that you define and must be unique within a Liberty collective. The status of a Liberty server cluster is stored in the collective repository.

A Liberty server that is configured into a collective can join a cluster by enabling the proper feature and configuring the cluster name. All members that specify the same cluster name are members of that cluster. The recommendation for a replica set is that it contains at least three collective controllers. A web server plug-in is used to distribute work across the servers in the cluster.

## Clustering in a collective

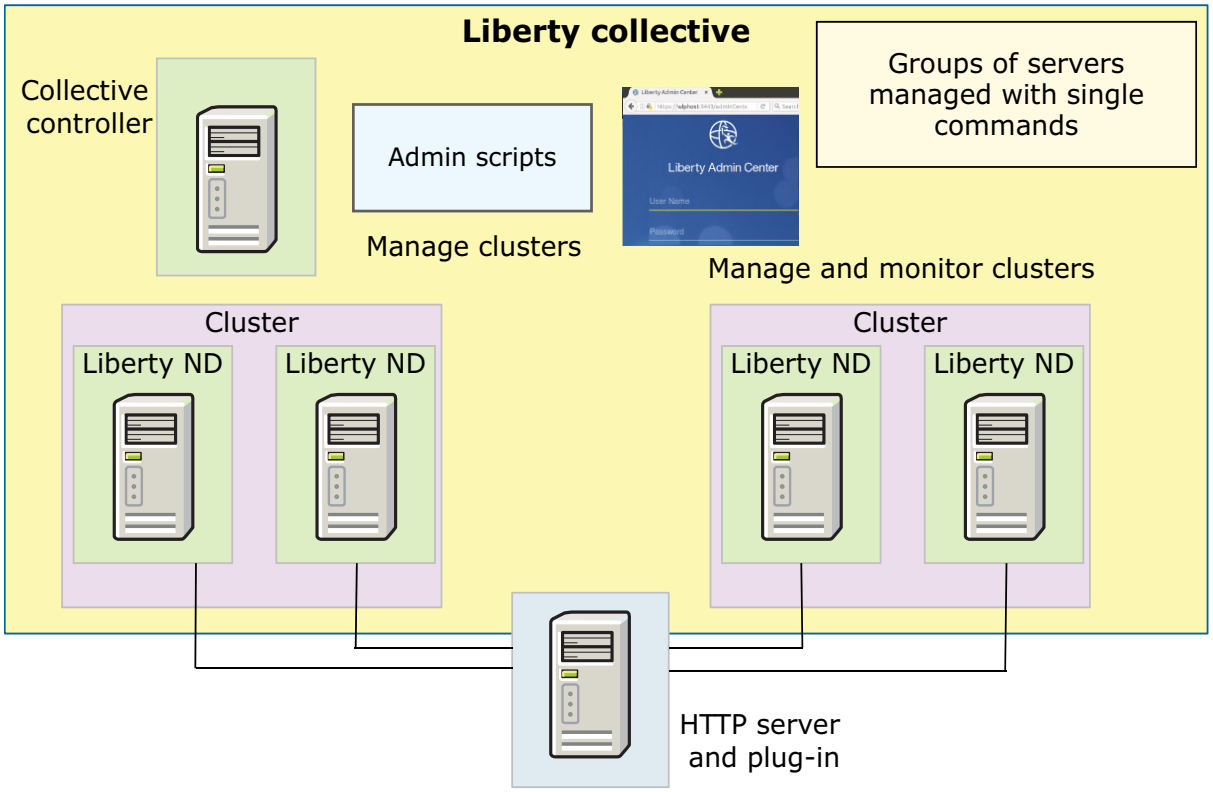






*Figure 2-30. Clustering in a collective*

Note that a Liberty collective is not a cluster; there is an important distinction. A collective is a formal grouping that is used to facilitate centralized administration of Liberty profile servers. A collective does not by itself provide clustering. You can have multiple clusters in a collective, but servers can belong to one cluster only.

The collective controller provides support for managing the servers in the cluster as one object, including starting and stopping the servers, updating the configuration of the servers, installing and uninstalling applications. The collective controller also provides you the capability of adding capacity to an existing cluster and generating the merged web server plug-in configuration for the cluster.

To use the cluster capabilities, the administrator builds a collective that consists of one Liberty server that acts as a collective controller and four Liberty servers to act as collective members. If failover of the controller itself is required (for highly available central management), a replica set of three or more controllers can be used (always an odd number).

Then, the collective members are configured as the application cluster. The time that is required for this is only a few minutes. Two simple commands create and configure a server as a collective controller or member. A server becomes a part of a cluster with the addition of the `clusterMember-1.0` feature and naming the same cluster name in a `clusterMember` element. Servers can be dynamically added to and removed from the cluster by updating their configuration.

In this scenario, the configuration for the application is stored in a common location and that file is pointed to with an include element in each server configuration file, thus deploying the application to the cluster. The administrator generates a web server plug-in that is used by the web server to route requests among the cluster members. When the web server receives requests for the application, the plug-in routes the requests to the servers in a round-robin manner.

# Example of server clusters and a Liberty collective

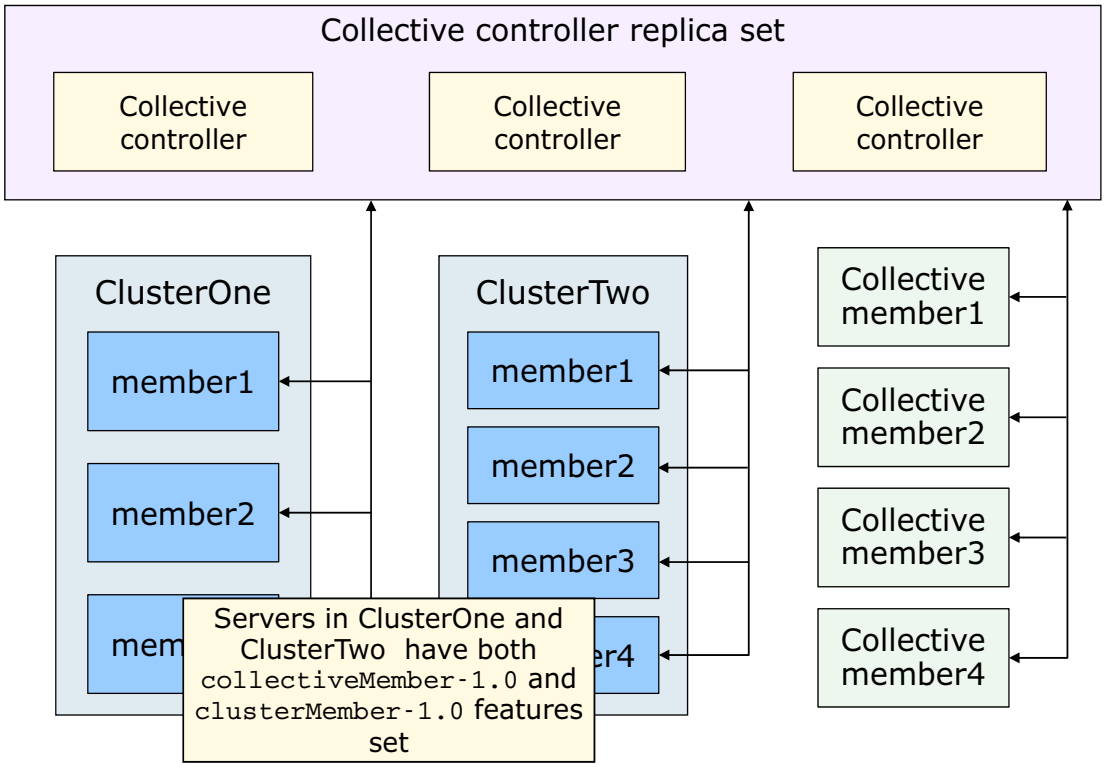






*Figure 2-31. Example of server clusters and a Liberty collective*

This diagram demonstrates a collective that has 11 collective members. Among the 11 members, 3 members belong to the server cluster named “ClusterOne”, four of them belong to the server cluster named “ClusterTwo”, and four do not belong to any server cluster.

You must define a Liberty cluster within a Liberty collective.

All Liberty cluster members must be members of the same collective.

# Create a Liberty cluster

- To create a Liberty cluster, you use the Liberty collective feature
  - Create a Liberty collective with at least two collective members
  - Set up the cluster
- To achieve a higher level of availability for your environment, you can configure a replica set with more controllers

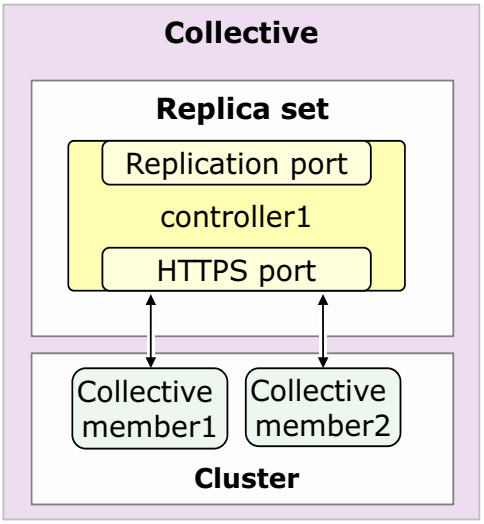






*Figure 2-32. Create a Liberty cluster*

To create a Liberty cluster, you use the Liberty `collective` feature.

First, create a Liberty collective with at least two collective members and then set up the cluster.

To be part of a server cluster, a Liberty profile server must be a member of a Liberty collective and also be configured with the `clusterMember` feature.

The replica set is logically present even when only one controller is in use.

The replication port is not the http or https port of the collective controller, but it is a unique port that is used for communication between the replicas of the replica set.

IBM Training

## Create a second collective member

- Create another collective member called **member2** by using the `server create` command
- Join the member to the collective by using the `collective join` command
- Update the `server.xml` file for **member2**
  - Add the collective join output
  - Enter unique HTTP and HTTPS ports (if member2 is on the same host)
- Start the new server **member2**
- Verify that the server starts correctly by viewing the `messages.log` file

```
CWWKX8112I: The server's host information was successfully
  published to the collective repository.
CWWKX8114I: The server's paths were successfully published to the
  collective repository.
CWWKX8116I: The server STARTED state was successfully published to
  the collective repository.
```

Multi-server management



*Figure 2-33. Create a second collective member*

Create another collective member called member2 by using the `server create` command; then join the member to the collective by using the `collective join` command.

Update the `server.xml` file for member2 by adding the collective join output. If member2 is on the same host as other Liberty servers, enter unique HTTP and HTTPS ports. Start the new server member2 and check the log messages to verify that it starts correctly.

IBM Training

## Adding collective members to the default cluster

- The default name of the first cluster in a collective is `defaultCluster`
- You can change the cluster name by specifying a name in the `clusterMember` name attribute when:
  - You have more than one cluster in the collective
  - You need a custom cluster name
- You can have multiple clusters in a collective but each member can be part of only one cluster at a time
- Add **member1** and **member2** to the default cluster by adding the following configuration element to the `server.xml` of each server:

```
<featureManager>
  <feature>clusterMember-1.0</feature>
</featureManager>
<clusterMember name="myCluster" />
```




*Figure 2-34. Adding collective members to the default cluster*

The default name of the first cluster in a collective is defaultCluster.

Add member1 and member2 to the default cluster by adding the clusterMember element to the `server.xml` of each server.

The cluster configuration is dynamically updated and published to the collective controller.

The servers member1 and member2 now belong to the cluster group named defaultCluster. Multiple clusters can be defined with a single collective, but a server can belong only to one cluster group at a time.

IBM Training

## Verify controller messages log

- The cluster configuration is dynamically updated and published to the collective controller
- Verify that the servers were added to the cluster by viewing the `messages.log` file of the controller
  - *<wlp_root>*`/usr/servers/controller1/logs/messages.log`
- Look for the following messages:

```
CWWKX9053I: The defaultCluster cluster has been created.

CWWKX9051I: The member1 server has been added to the
  defaultCluster cluster.

CWWKX9051I: The member2 server has been added to the
  defaultCluster cluster.
```




*Figure 2-35. Verify controller messages log*

A collective server can be configured to be a cluster member while it is running. The new configuration is updated dynamically and published to the collective controller. If a collective server is configured to be a cluster member while the server is stopped, the server does not become a member of a cluster until the server is started.

Verify that the servers were added to the cluster by viewing the `messages.log` file of controller. Look for messages that indicate that the cluster is created and the members are added to the cluster.

```
[9/9/16 6:51:05:867 EDT] 0000002e
bm.ws.collective.repository.internal.ClusterManagerMBeanImpl I
CWWKX9053I: The defaultCluster cluster has been created.
```

```
[9/9/16 6:51:05:868 EDT] 0000002e
bm.ws.collective.repository.internal.ClusterManagerMBeanImpl I
CWWKX9051I: The member1 server has been added to the defaultCluster
cluster.
```

```
[9/9/16 6:52:19:481 EDT] 0000002e
bm.ws.collective.repository.internal.ClusterManagerMBeanImpl I
CWWKX9051I: The member2 server has been added to the defaultCluster
cluster.
```

# 2.4. Liberty Admin Center

IBM Training

# Liberty Admin Center




*Figure 2-36. Liberty Admin Center*

IBM Training

## Topics

- Liberty collective management model
- Creating a Liberty collective
- Setting up a Liberty cluster
- Liberty Admin Center
- Alternative Liberty management models in cloud environments




*Figure 2-37. Topics*

IBM Training

## Stand-alone server management

- Admin Center on a stand-alone server can manage only that server

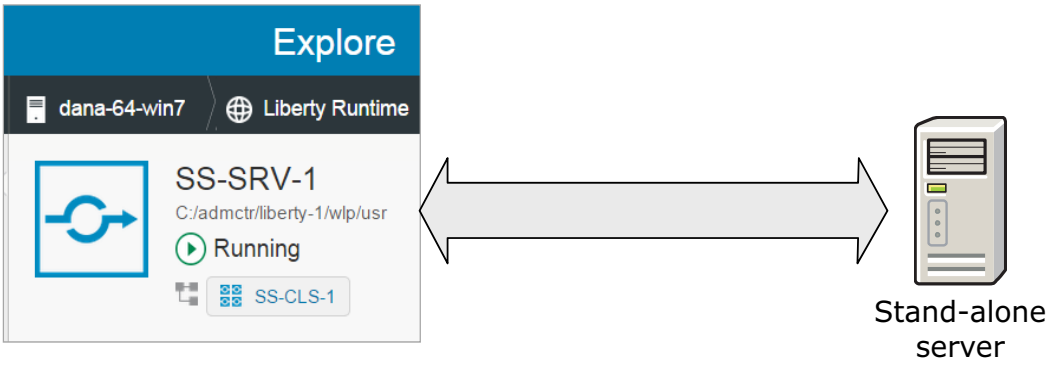






*Figure 2-38. Stand-alone server management*

The Admin Center feature can be added to stand-alone servers so that it can manage only that server.

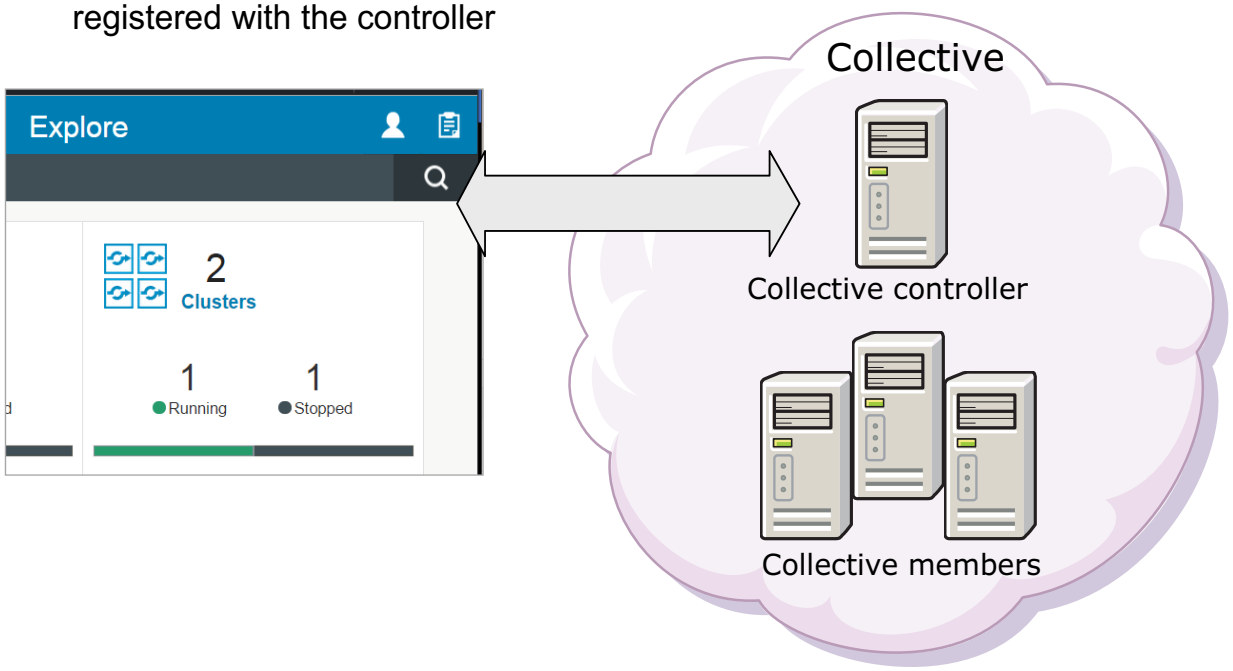
The following statement shows the Admin Center feature that is included in the stand-alone `server.xml` file:

```
<featureManager>
        <feature>jsp-2.2</feature>
        <feature>adminCenter-1.0</feature>
 </featureManager>
```

IBM Training

## Management of collectives

- Admin Center on the collective controller
  - Can manage all servers in the collective
  - Deploy server package archives to hosts registered with the controller




Multi-server management



*Figure 2-39. Management of collectives*

The Admin Center feature can be added to a collective controller. Then, it can manage all servers in the collective.

You can deploy server package archives to any host that is registered with the controller.

IBM Training

# Admin Center Toolbox

- Access the Admin Center with the URL
  `http://<hostname>:9080/adminCenter`

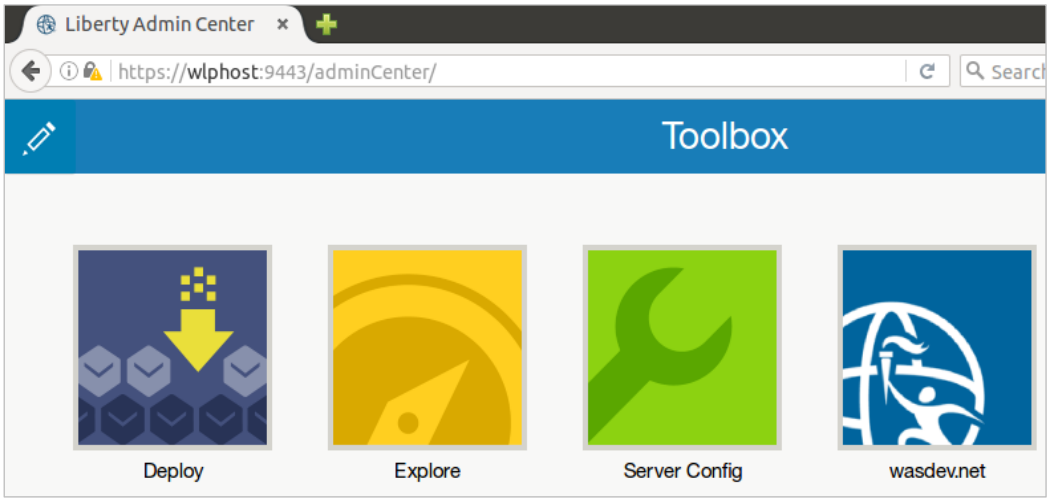






*Figure 2-40. Admin Center Toolbox*

Log in to the Admin Center by using the name and password that you specified for `<quickStartSecurity>`. After logging in, you see the toolbox view and some tool icons.

The **Explore** tool is how you examine the Liberty profile topology. You can view the status of servers and applications. You can also start, stop, or restart servers and applications.

The **Deploy** tool is how you install Liberty profile server package (archive) files on registered hosts within the collective.

## Admin Center: Servers view

- Log in to the Admin Center and view **Explore > Servers**
- With two collective members, you can proceed to create a cluster

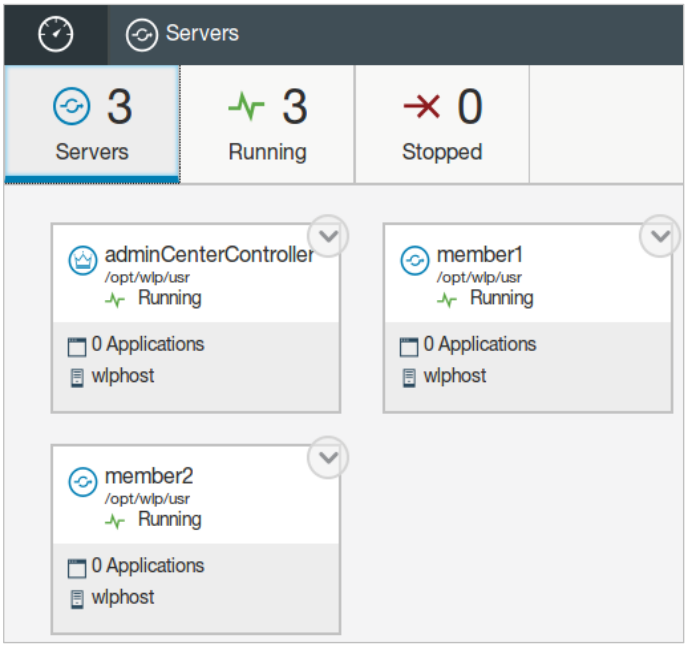






*Figure 2-41. Admin Center: Servers view*

The Servers view in the Admin Center shows the state of all three servers in the current collective. The controller and member1 and member2 are all running.

IBM Training

# Admin Center: Clusters view

- You can start, restart, or stop a cluster from the Admin Center

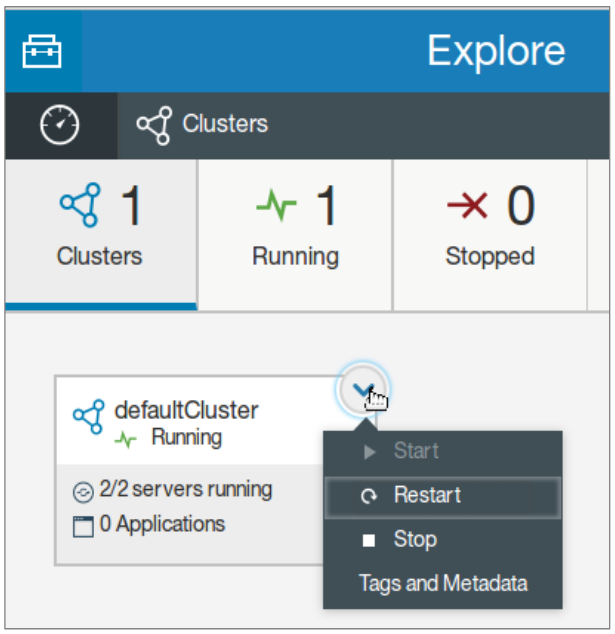






*Figure 2-42. Admin Center: Clusters view*

The All Clusters view in the Admin Center shows the status of all clusters, which can be started, partially started, or stopped.

Individual clusters names are shown. The state of the cluster members and the number of applications is also shown.

You can use this view to complete actions on the cluster as shown in the screen capture.

Any action that is completed on the cluster, such as stop, start, and restart, applies to all the cluster members.

# 2.5. Alternative Liberty management models in cloud environments

IBM Training

# Alternative Liberty management models in cloud environments




*Figure 2-43. Alternative Liberty management models in cloud environments*

IBM Training

## Topics

- Liberty collective management model
- Creating a Liberty collective
- Setting up a Liberty cluster
- Liberty Admin Center
- Alternative Liberty management models in cloud environments

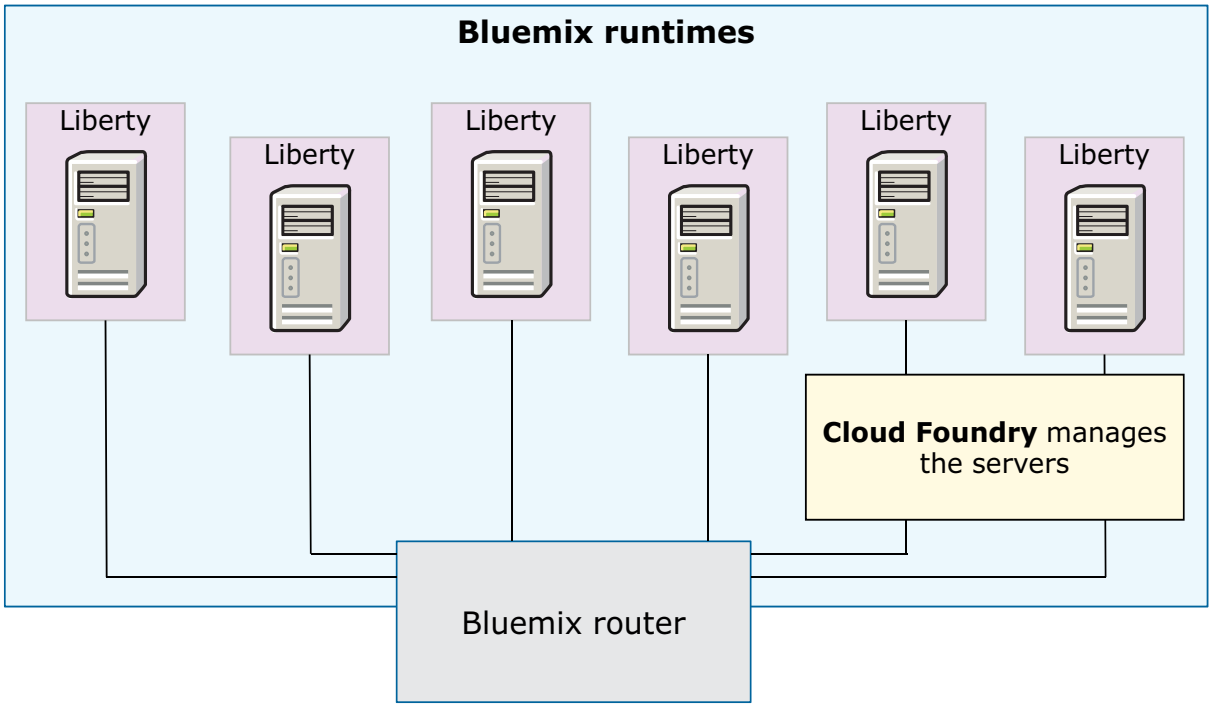



*Figure 2-44. Topics*

Liberty provides a common runtime environment for applications that are running in different cloud environments. However, the administration of Liberty in these different environments tends to vary widely. At one end of the spectrum, a PaaS environment can provide a completely installed, configured, and managed runtime, with little for an administrator to do other than monitor the health of the application. At the other end of the spectrum, an infrastructure as a service (IaaS) environment provides a hosted virtual machine (VM), and the administrator needs to install, configure, and manage the runtime environment in much the same way as in a data center. The cloud provides a great deal of choice, and the degree of administrative involvement and control is often a key factor in deciding which type of cloud service to use. Administrative operations and responsibilities increase as you move from Instant Runtimes (Cloud Foundry1), Containers (Docker2), and Bluemix virtual machines on OpenStack3, to IBM WebSphere Application Server on Cloud. You can choose the best infrastructure for your environment, or use one of these products in combination with your current application, data, and services.

IBM Training

# Put the servers in a Platform as a Service (PaaS)




Multi-server management



*Figure 2-45. Put the servers in a Platform as a Service (PaaS)*

In the case where servers are running on the Bluemix Platform-as-a-service, Cloud Foundry manages the servers.

In platform as a service (PaaS), everything is provided except the application code, users, and data. Typically, when using a PaaS, the vendor maintains the application server, databases, and all of the necessary operating system components. PaaS provides a complete environment that is actively managed. For most runtimes, PaaS provides scaling without modification.

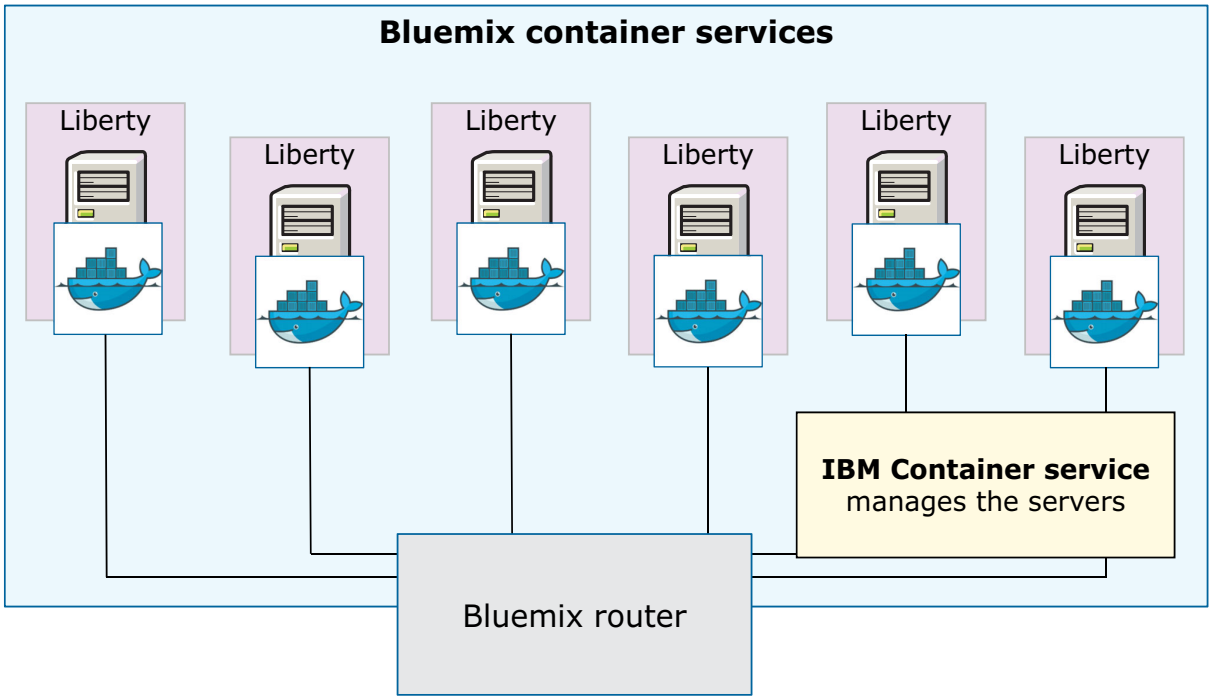
Instant Runtimes is a PaaS service that is provided by Bluemix. You can use runtimes to get your application up and running quickly, with no need to set up and manage VMs and operating systems. Instant Runtimes is based on Cloud Foundry, which means that community buildpack or tooling plug-ins for Cloud Foundry also work with Instant Runtimes. Bluemix Instant Runtimes provides a dashboard for you to create, view, and manage your applications and services, and to monitor application resource usage. With the Bluemix dashboard, you can also manage organizations, spaces, and user access. It provides access to a wide variety of services that can be incorporated into an application.

With Bluemix Instant Runtimes, you can push Java EE applications (WAR or EAR file formats) and select what services are needed for your application. The Liberty buildpack generates the server configuration. However, you might need to modify the `server.xml` file for a standard Liberty deployment. For this and similar cases, you need to package your Liberty server and push the

package to Bluemix instead of the WAR or EAR file. This package includes everything on the Liberty server, including the modified `server.xml` file and your application. In Instant Runtimes, no direct involvement of an administrator is needed.

IBM Training

# Put the servers in containers




Multi-server management



*Figure 2-46. Put the servers in containers*

IBM Container services can manage the servers when the servers are running in Docker containers on the Bluemix cloud.

You can use containers to package an application and all of its dependencies into a standardized unit for software development and deployment. Containers can enable applications to run reliably and easily when moved from one computing environment to another. Containers guarantee that applications that are run in the same way, regardless of the environment the container is running in. This can be from a developer's computer to a test environment, from a staging environment to production, or from a physical machine in a data center to a VM in a private or public cloud. Every container runs applications in an isolated environment. Moreover, the isolation allows you to run many containers simultaneously on one host.

## Review questions

1. True or False: The collective create command establishes the administrative domain security configuration.
2. True or False: When you start a collective controller, you see the following message in the log file: `The CollectiveRepository MBean is available.`
3. True or False: The collective replicate command is used to join a member to the collective.
4. True or False: The Admin Center feature is part of the default template for all Liberty servers.
5. True or False: Liberty profile automatically creates clusters of Liberty servers when you create a collective.
6. True or False: You can restart a cluster from the Admin Center.
7. True or False: The default name of the first cluster in a collective is defaultCluster.
8. True or False: You can have only one cluster in a collective.




*Figure 2-47. Review questions*

Write your answers here:

1.

2.

3.

4.

5.

6.

7.

8.

IBM Training

## Review answers (1 of 2)

1. True or False: The collective create command establishes the administrative domain security configuration.
   The answer is True.
2. True or False: When you start a collective controller, you see the following message in the log file: `The CollectiveRepository MBean is available.`
   The answer is True.
3. True or False: The collective replicate command is used to join a member to the collective.
   The answer is False.
   The collective join command is used to join a member to the collective.
4. True or False: The Admin Center feature is part of the default template for all Liberty servers.
   The answer is False.
   You must add the Admin Center feature to the `server.xml` file of a Liberty server.




*Figure 2-48. Review answers (1 of 2)*

## Review answers (2 of 2)

5. True or False: Liberty profile automatically creates clusters of Liberty servers when you create a collective
   The answer is False.
   Server clusters are added by using the clusterMember-1.0 feature.
6. True or False: You can restart a cluster from the Admin Center.
   The answer is True.
7. True or False: The default name of the first cluster in a collective is defaultCluster.
   The answer is True.
8. True or False: You can have only one cluster in a collective.
   The answer is False.
   You can have multiple clusters in a collective, but each collective member can be part of only one cluster at a time.




*Figure 2-49. Review answers (2 of 2)*

## Unit summary

- Describe the Liberty collective architecture
- Describe how to create and configure a Liberty collective
- Describe the use of clusters in a collective
- Describe how to configure and manage a Liberty server cluster
- Describe how the Admin Center works with different topologies
- Describe the ways that Liberty servers can be managed in cloud environments




*Figure 2-50. Unit summary*

IBM Training

# Exercise: Managing Liberty collectives with the Admin Center




*Figure 2-51. Exercise: Managing Liberty collectives with the Admin Center*

IBM Training

## Exercise objectives

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




*Figure 2-52. Exercise objectives*

# Unit 3. Administration and application deployment with scripting

## Estimated time

00:45

## Overview

This unit describes how to use administration scripts to manage collectives. You learn how to deploy applications to Liberty servers.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

IBM Training

# Unit objectives

- Describe the Liberty Management API
- Describe the installation of a Java-enabled runtime for scripting
- Describe the management of a collective with Jython scripts
- Describe how to deploy an application to a Liberty cluster
- Describe ways to generate the HTTP plug-in configuration for a cluster




*Figure 3-1. Unit objectives*

IBM Training

## Simple and flexible management API

- Liberty Management API
  - JMX MBeans
  - REST API
    - Mapping to MBeans
    - REST-only – File Transfer

Ways to call the Liberty Management API:
**Graphically** with the Admin Center UI
or
**Scripting** with Java or REST commands

- Admin Center web-based UI
  - Enabled on any server with the `adminCenter-1.0` feature

- Scripting
  - Any Java-enabled language (Jython, JRuby, Groovy, and other languages)
  - Any REST-capable language (Python, cURL, Go, and other languages)
  - Liberty does not include a script language runtime
  - Many samples on IBM developerWorks (wasdev.net)




*Figure 3-2. Simple and flexible management API*

The management API of Liberty is based on the use of JMX. JMX is a framework that provides a standard way of exposing Java resources, in this case the Liberty servers, to JMX-enabled clients. An increasing number of Representational State Transfer (REST)-based interfaces are being created to manage Liberty in addition to the JMX interfaces. Various tools can be used to connect to the JMX framework.

JMX is an API that is used to set up applications for management and monitoring. The API is included in Java Standard Edition (J2SE) version 5 and later.

Liberty servers are administered by using line commands, scripts, the Admin Center web UI, APIs, or from the WebSphere developer tools. Liberty Management APIs include Java Management Extensions (JMX) MBeans and Representational State Transfer (REST) application programming interfaces (APIs). Liberty can be administered through JMX calls to the server MBeans and by direct modification or replacement of the configuration files; no command-line tool is required. REST APIs can be used for mapping to MBeans or for file transfer.

The WebSphere Liberty Administrative Center (Admin Center) feature is available for Liberty, which provides a web-based graphical interface for Liberty servers and resource management. It is designed around a toolbox model, so you can select tools in a customized Admin Center instance. Do not confuse this with the Integration Solutions Console, also called the Admin Console, in

WebSphere Application Server. The Liberty Admin Center is designed to be more goal and task oriented.

Liberty does not ship a scripting language runtime. However, for simple and flexible management by using scripting, you can use the following items:

- Any Java enabled language such as Jython, JRuby, Groovy, or others
- Any REST-capable language such as Python, CURL, Go, or others
- Various sample scripts on WASdev.net

On a z/OS platform, you can use IBM MVS operator commands to start, stop, or modify Liberty. Liberty servers can also be accessed from JMX clients, such as using the jConsole tool provided in the Java SDK to monitor data.

IBM Training

## What are JMX MBeans?

- **MBeans** are *managed beans*
  - Java objects that represent resources that can be managed
- An MBean has a management interface that consists of:
  - Named and typed *attributes* that can be read and written
  - *Operations* that can be invoked
  - Typed *notifications* that the MBean can emit
- A standard MBean is a Java object that conforms to the JavaBeans component model
  - Attributes are exposed through "getter" and "setter" methods
  - All methods are defined statically in the MBean interface
- A dynamic MBean defines its management interface at run time
  - Determines the names and types of the attributes by parsing an XML file
- All internal Liberty MBeans follow the ModelMBean pattern
  - Uses an XML descriptor and creates an instance of a ModelMBean that matches the descriptor




*Figure 3-3. What are JMX MBeans?*

JMX MBeans are *managed beans.* To be useful, an MBean must be registered in an MBean server. Starting with release 5.0, J2SE includes a built-in *platform MBean server*.

All of the internal WebSphere MBeans follow the Model MBean pattern. Pure Java classes supply the real logic for management functions, and the WebSphere MBeanFactory class reads the description of these functions from the XML MBean Descriptor and creates an instance of a ModelMBean that matches the descriptor. This ModelMBean instance is bound to your Java classes and registered with the MBeanServer running in the same process as your classes.

IBM Training

## Jython runtime

- Liberty does not include a scripting runtime
  - Developers can choose a Java-enabled scripting runtime
  - Jython is used as the scripting language in the course exercises
- Jython is an implementation of Python, which is written entirely in Java
- A Java runtime is needed to install and run Jython
- Install Jython into a directory under the Liberty runtime and add the directory to the system path
- To install Jython v2.5.3, enter the following command:
  - `java -jar jython-installer-2.5.3.jar`




*Figure 3-4. Jython runtime*

Liberty ships with 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 has been tested with Jython and is compatible with versions 2.5.4 and higher. Liberty does not ship with a Jython run time so the user needs to download and configure the run time for themselves.

To use the Jython scripting capabilities, the Jython environment needs to be set up.

Jython is distributed as an executable JAR file installer. After you have the JAR file, either double-click the `jython_installer-X.jar` or run java with the `-jar` option.

For installation instructions, see https://wiki.python.org/jython/InstallationInstructions.

Developers who have used WebSphere Application Server V8.5.5 are likely to be familiar with Jython since the wsadmin tool of WebSphere Application Server uses Jython V2.1.

# Jython installation

- Choose the language
- Accept the license terms
- Choose the installation type, target directory, Java runtime

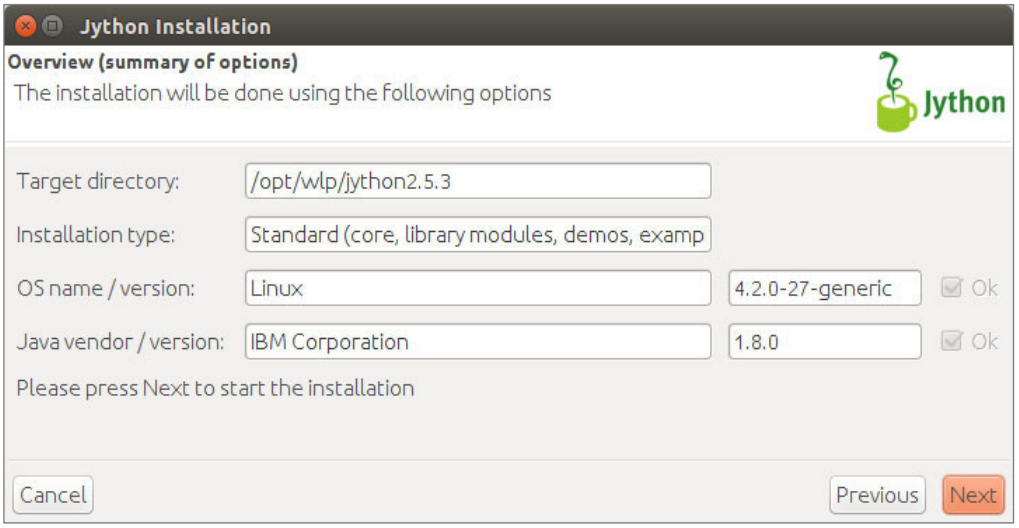


*Figure 3-5. Jython installation*

The installer walks through a similar set of steps in graphical or console mode: showing the license, selecting an install directory and JVM, and copying Jython to the file system. After this completes, Jython is installed in the directory that you selected. There is a script in the install directory, Jython on Unix like systems or `jython.bat` on Windows, that starts up the Jython console which can be used to dynamically explore Jython and the Java runtime.

IBM Training

## ClusterManager MBean

- You can use JConsole or Jython scripts to access the MBeans
- The ClusterManager MBean enables the collective controller to act as an operational repository
- Perform certain data queries against the operational cache within the collective controller such as:
  - List cluster name
  - List members
  - Get status
- Download the following Jython scripts from the Liberty Repository
  - `listClusterNames.py`
  - `listClusterMembers.py`
  - `getClusterStatus.py`




*Figure 3-6. ClusterManager MBean*

You can use Jython-based scripts to establish a Java Management Extensions (JMX) MBean Liberty server connection.

The ClusterManager MBean is an example of how the collective controller acts as an operational repository. Data queries, such as the `listClusterNames`, `listMembers`, and `getStatus` operations are performed entirely against the operational cache within the collective controller. Operations that require action on a target such as start and stop are performed against the respective collective member.

# IBM WebSphere Liberty Repository

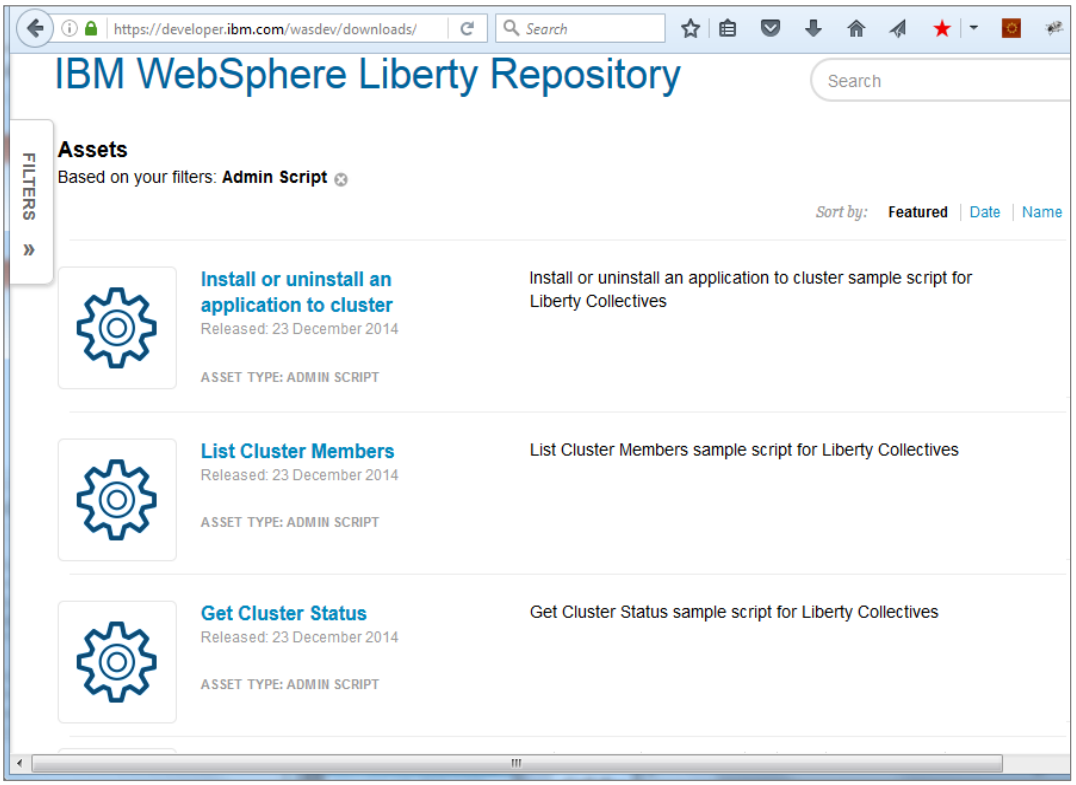






*Figure 3-7. IBM WebSphere Liberty Repository*

You can use the Liberty Repository to easily extend or enhance your Liberty-based applications.

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. These assets are specifically designed to encompass end-to-end integration and provide important business value for the entire lifecycle of your Liberty application.

There are 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.

Filter the assets on the repository to display all admin scripts.

# Liberty Repository sample

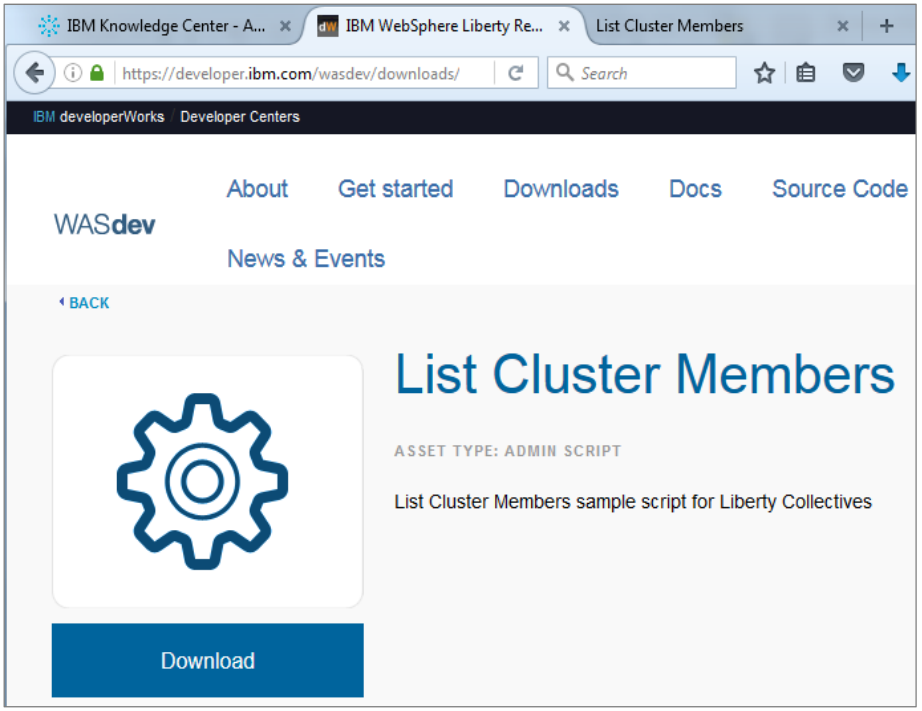






*Figure 3-8. Liberty Repository sample*

Developers can download sample scripts from the IBM WebSphere Liberty Repository and use them in their own environments.

## Sample: listClusterMembers.py script

- The `listClusterMembers.py` sample demonstrates how to call the ClusterManager MBean from a Jython script to list all the cluster members in a collective.
- The ClusterManager MBean operation used:
  - listMembers




*Figure 3-9. Sample: listClusterMembers.py script*

The `listClusterMembers` sample script lists the members of a static cluster with the given cluster name. A connection is made to the collective controller located at the given host and port, and the request to list the cluster members is passed to the ClusterManager MBean.

# Required parameters: listClusterMembers.py

| Parameter | Description |
|---|---|
| <first parameter> | The cluster name |
| --truststore | The path to the truststore to be used when establishing a connection to the collective controller |
| --truststorePassword | The password for the truststore that is specified by the --truststore parameter |
| --host | The host name where the collective controller is running |
| --port | The https port where the collective controller is listening |
| --user | The user name to use when connecting to the collective controller |
| --password | The password to use when connecting to the collective controller |




*Figure 3-10. Required parameters: listClusterMembers.py*

The table shows the parameters that are required by the `listClusterMembers.py` script.

The first parameter is the cluster name. The remaining parameters include the parameter name and its associated value.

IBM Training

## Sample use and resulting messages

- Run the script from a terminal window

```
$> jython /opt/labfiles/management/listClusterMembers.py
defaultCluster --host=wlphost --port=9443 --user=admin --
password=passw0rd --
truststore=/opt/wlp/usr/servers/controller1/resources/security
/trust.jks --truststorePassword=passw0rd

Connecting to the server...
Successfully connected to the server "wlphost:9443"
The following members exist in cluster defaultCluster
wlphost,/opt/wlp/usr,member1
wlphost,/opt/wlp/usr,member2
```




*Figure 3-11. Sample use and resulting messages*

A sample script file and result is displayed:

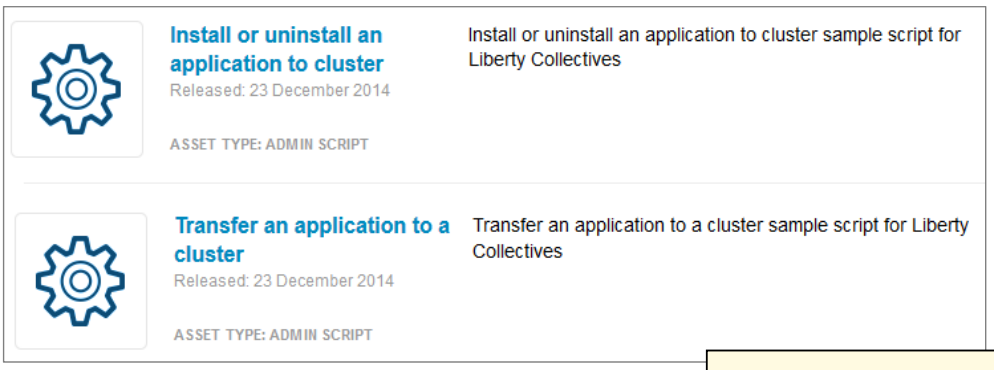
**Example**

```
jython /opt/labfiles/management/listClusterMembers.py defaultCluster
--host=wlphost --port=9443 --user=admin --password=passw0rd
--truststore=/opt/wlp/usr/servers/controller1/resources/security/trust.j
ks --truststorePassword=passw0rd

Connecting to the server...

Successfully connected to the server "wlphost:9443"

The following members exist in cluster defaultCluster

wlphost,/opt/wlp/usr,member1

wlphost,/opt/wlp/usr,member2
```

IBM Training

# How to deploy applications to a Liberty cluster

- Use sample scripts that are available from the Liberty Repository to deploy applications and configuration to all members of a cluster




- Scripts include
  - Install or uninstall an application to a cluster
  - Transfer an application to a cluster

The script that comes with **Install or Uninstall an application to a cluster** is named **manageAppOnCluster.py**




*Figure 3-12. How to deploy applications to a Liberty cluster*

You can use sample scripts that are available from the Liberty Repository to deploy applications and configuration to all members of a cluster.

Several administrative scripts are available for download from the Liberty Repository. Many of these scripts are written in Jython. Other scripts are written in JRuby or Groovy. Jython is not distributed with any edition of the Liberty Profile, so you must download it from the Jython organization website and install it.

The required parameters and optional parameters for each script are documented in the Liberty Repository.

IBM Training

## Install or uninstall a simple application to a cluster

- The `manageAppOnCluster.py` script is used to install a simple application to all members of a specific cluster
  - Is used to install or uninstall only a WAR application file
- The script does the following tasks
  - Connects to the collective controller
  - Obtains the list of cluster members by using the ClusterManager MBean
  - Passes a request to the FileTransfer MBean to upload or delete the application file
- The script also adds the application element to, or removes the application element from, the `server.xml` of each cluster member
  - By default, this script installs WAR files into `${server.config.dir}/apps` unless the `--appDir` option is used




*Figure 3-13. Install or uninstall a simple application to a cluster*

The `manageAppOnCluster.py` script is used to install a simple application to all members of a specific cluster. This script is used to install or uninstall only a WAR application file.

The script also adds the application element to, or removes the application element from, the `server.xml` of each cluster member. By default, this script installs WAR files into the apps directory unless the `-appDir` option is used to specify a different directory.

Because this script uses the FileTransfer MBean to upload one or more files, the remoteFileAccess element must be specified in the `server.xml` of the server that receives the file. Without this, you get a file permission error when using the script.

---

1+1=2 **Example**

Here is an example of a remoteFileAccess element:

```
<remoteFileAccess>
<writeDir>${server.config.dir}</writeDir>
</remoteFileAccess>
```

---

IBM Training

# Using the manageAppOnCluster script

- Example of how to install an application:
  - `<Jython_root>/jython manageAppOnCluster.py --install=/opt/applications/snoop.war --truststore=<wlp_root>/usr/servers/controller1/resources/security/trust.jks --truststorePassword=password --host=localhost --port=9443 --user=admin --password=adminpwd --clusterName=defaultCluster`

```
Connecting to the server...
Successfully connected to the server "localhost:9443"
Uploading application snoop.war to localhost,/opt/wlp/usr,member1
Updating server config for localhost,/opt/wlp/usr,member1
Complete
Uploading application snoop.war to localhost,/opt/wlp/usr,member2
Updating server config for localhost,/wlp/usr,member2
Complete
```




*Figure 3-14. Using the manageAppOnCluster script*

This example shows the required parameters for the `manageAppOnCluster` script.

The `-install` parameter specifies the file system path of the application WAR file to install. Use the `-uninstall` parameter to remove the WAR file.

Look for messages similar to the ones shown on this slide to verify that the installation or uninstall is successful.

IBM Training

# Web server plug-in communication

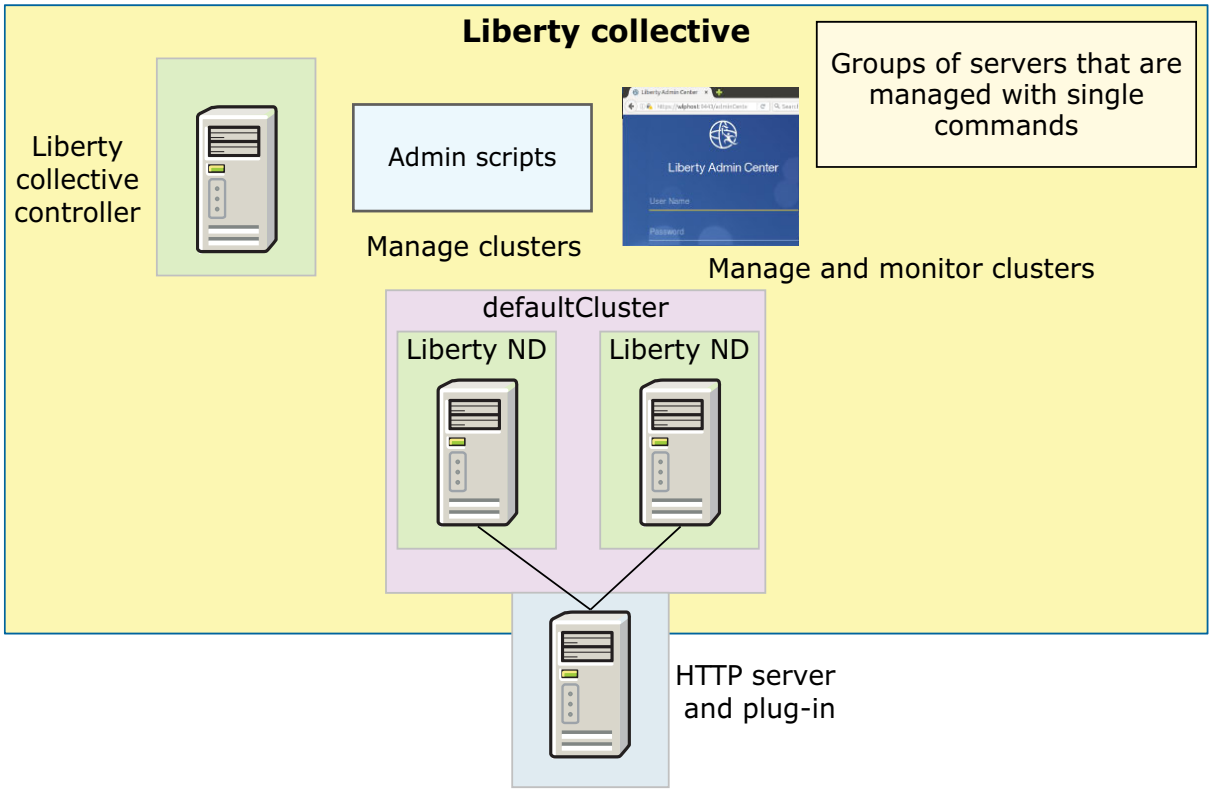






*Figure 3-15. Web server plug-in communication*

You can configure a web server plug-in to receive an HTTP request for dynamic resources. You can forward the request to the Liberty server, which provides high-availability and workload balancing through the web server plug-in.

Web server plug-ins enable the web server to communicate requests for dynamic content, such as servlets, to the application server. A web server plug-in is associated with each web server definition. The configuration file (`plugin-cfg.xml`) that is generated for each plug-in is based on the applications that are routed through the associated web server.

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 a Liberty server. The web server plug-in then uses this information to forward the request to the Liberty server.

IBM Training

# Generate HTTP plug-in configuration for the cluster (1 of 4)

- Two ways to generate the plug-in configuration
  - The `genClusterPlugin.py` script
  - JConsole

- Use the following format to run the `genClusterPlugin.py` script:
  - *<Jython_root>*`/jython genClusterPlugin.py defaultCluster --host=wlphost --port=9443 --user=admin --password=adminpwd --truststorePassword=tsPassword --truststore=/opt/wlp/usr/servers/controller1/resources/security/trust.jks`

```
Connecting to the server...
Successfully connected to the server “wlphost:9443"
Generated plugin-cfg.xml file:
  /opt/wlp/servers/controller1/pluginConfig/defaultCluster-plugin-
  cfg.xml
This file will reside in the host file system of the controller.
```




*Figure 3-16. Generate HTTP plug-in configuration for the cluster (1 of 4)*

There are a number of ways to generate the plug-in configuration. You can use the `genClusterPlugin.py` script or the JConsole tool.

An example of how to use the `genClusterPlugin.py` script is shown on this slide. As you can see from the output that is shown, the generated file resides in the host file system of the controller.

IBM Training

# Generate HTTP plug-in configuration for the cluster (2 of 4)

- You can also use JConsole (Java Monitoring and Management Console) to generate the HTTP plugin-configuration file
- When using JConsole, perform the following tasks
  - Connect to the collective controller
  - Select the ClusterManager MBean
  - Select the `generateClusterPluginConfig` operation
- Start JConsole by entering `jconsole` at the command prompt
  - Your path must contain the bin directory of the SDK
  - An IBM SDK is installed with the IBM HTTP Server




*Figure 3-17. Generate HTTP plug-in configuration for the cluster (2 of 4)*

Alternatively, you can use the JConsole tool to invoke operations on the ClusterManager MBean that generates the cluster plug-in configuration.

Start JConsole by entering `jconsole` at the command prompt. Your path must contain the bin directory of the SDK.

In the exercises, the SDK was installed with the IBM HTTP Server as part of the course image.

Run the jconsole command from the `/opt/IBM/HTTPServer/java/8.0/bin` directory.

# Generate HTTP plug-in configuration for the cluster (3 of 4)

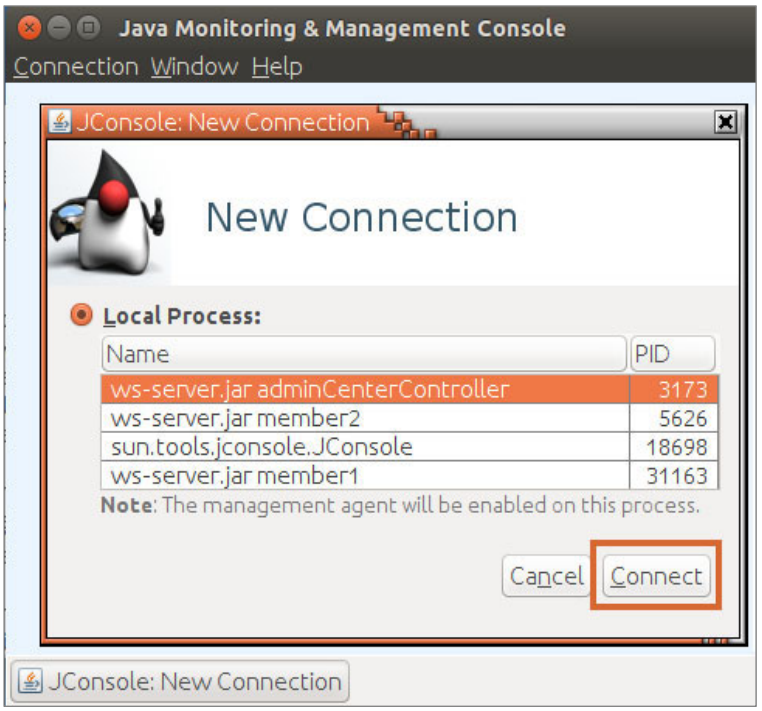






*Figure 3-18. Generate HTTP plug-in configuration for the cluster (3 of 4)*

When JConsole starts, select the collective controller in the list of local processes. Then, click **Connect**.

The JConsole application initially tries to make a secure connection to the Liberty server. Because no security has been configured on the Liberty server, the initial secure connection fails and you are then prompted to either cancel or create an insecure connection. Select the option to create an insecure connection

# Generate HTTP plug-in configuration for the cluster (4 of 4)

- With the collective controller connected, select the ClusterManager MBean, and the `generateClusterPluginConfig` operation

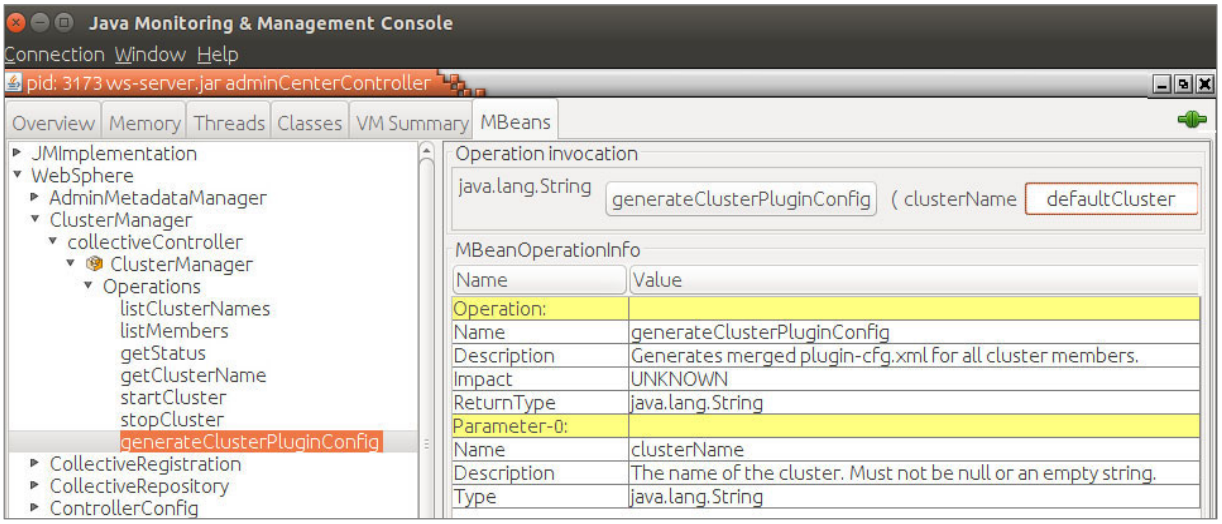


*Figure 3-19. Generate HTTP plug-in configuration for the cluster (4 of 4)*

JConsole uses the extensive instrumentation of the Java virtual machine running the Liberty server. JConsole uses this to provide information about the performance and resource consumption of the Liberty server itself and applications running within the server. Tabs are provided to show an overview of the resource usage. The last tab, the **MBean** tab shows the JMX resources that are available within the Liberty server runtime, and allows a user of the JConsole application to invoke the operations on these MBeans.

In summary, the local connector allows the use of JConsole to connect to a running Liberty server. The JConsole application allows the user to browse and invoke operations available on the MBeans provided by the Liberty server process.

With the collective controller connected, select the ClusterManager MBean, and the `generateClusterPluginConfig` operation. The screen capture shows several other useful operations such as startCluster, stopCluster, and getStatus.

The `generateClusterPluginConfig` operation returns the plug-in configuration file to the `pluginConfig` directory of the collective controller.

# Generated sample plug-in file

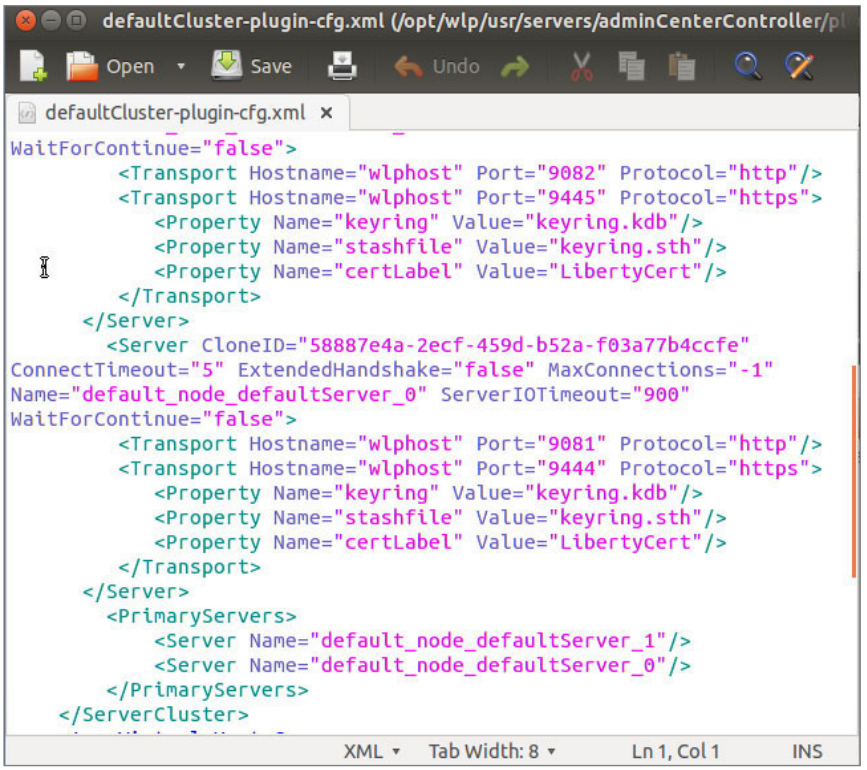






*Figure 3-20. Generated sample plug-in file*

The screen capture shows a section of the generated sample `defaultCluster-plugin-cfg.xml` file.

## Unit summary

- Describe the Liberty Management API
- Describe the installation of a Java-enabled runtime for scripting
- Describe the management of a collective with Jython scripts
- Describe how to deploy an application to a Liberty cluster
- Describe ways to generate the HTTP plug-in configuration for a cluster




*Figure 3-21. Unit summary*

IBM Training

## Review questions

1. True or False: The `manageAppOnCluster.py` script is used to install or uninstall only WAR application files.

2. True or False: The `genClusterPlugin.py` script generates the plug-in configuration file to the `pluginConfig` folder of the controller.




*Figure 3-22. Review questions*

Write your answers here:

1.

2.

## Review answers

1. True or False: The `manageAppOnCluster.py` script is used to install or uninstall only WAR application files.
   The answer is True.

2. True or False: The `genClusterPlugin.py` script generates the plug-in configuration file to the `pluginConfig` folder of the controller.
   The answer is True.




*Figure 3-23. Review answers*

IBM Training

# Exercise: WebSphere Liberty administration by using Jython Scripts




*Figure 3-24. Exercise: WebSphere Liberty administration by using Jython Scripts*

IBM Training

## Exercise objectives

- 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




*Figure 3-25. Exercise objectives*

# Unit 4. Dynamic Routing

## Estimated time

00:45

## Overview

In this unit, you learn how Dynamic Routing causes web requests to be routed successfully as the routing topology changes.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

IBM Training

## Unit objectives

- Describe routing by the web server when using the WebSphere plug-in
- Describe the purpose of Dynamic Routing
- Describe benefits of Dynamic Routing
- Describe the components that are used in Dynamic Routing
- Explain how to configure Dynamic Routing




*Figure 4-1. Unit objectives*

# 4.1. Web server routing with the WebSphere plug-in

IBM Training

# Web server routing with the WebSphere plug-in




*Figure 4-2. Web server routing with the WebSphere plug-in*

IBM Training

## Topics

- Web server routing with the WebSphere plug-in
  - Dynamic Routing overview
  - Configuring Dynamic Routing




*Figure 4-3. Topics*

# Web server plug-in routing (1 of 2)

- Web server with WebSphere plug-in routes requests
- Requests are routed based on the information in the plug-in configuration file
  - The file is generated by using Liberty utilities
  - Failed requests determine which servers and applications are unavailable

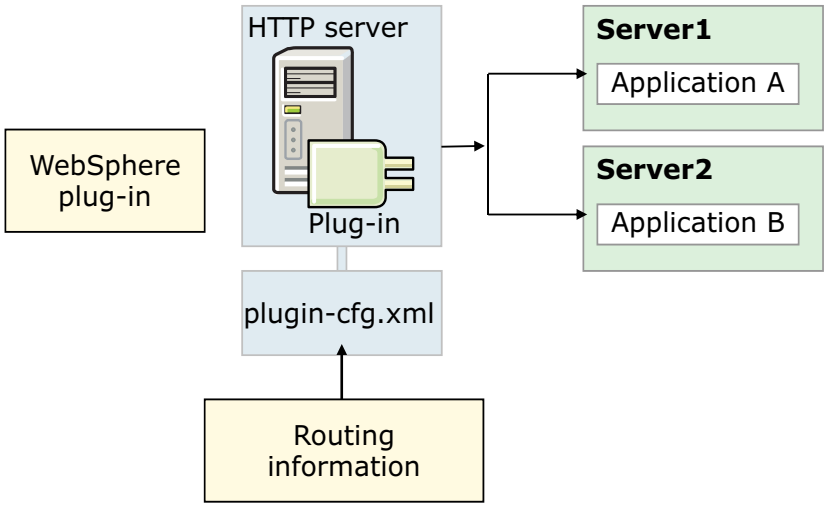






*Figure 4-4. Web server plug-in routing (1 of 2)*

Routing of web requests to application servers in a Liberty collective is done by using a web server with the WebSphere plug-in. With web server plug-in routing, the information that is used to route requests is read from a file. The routing information file contains the endpoint information of the servers in the collective (hosts, ports, and more), and the application information (URI's, virtual hosts, and more). 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.

# Web server plug-in routing (2 of 2)

- Routing information must be regenerated as the routing endpoints change
  - Servers added, removed, or updated
  - Applications added, removed, or updated

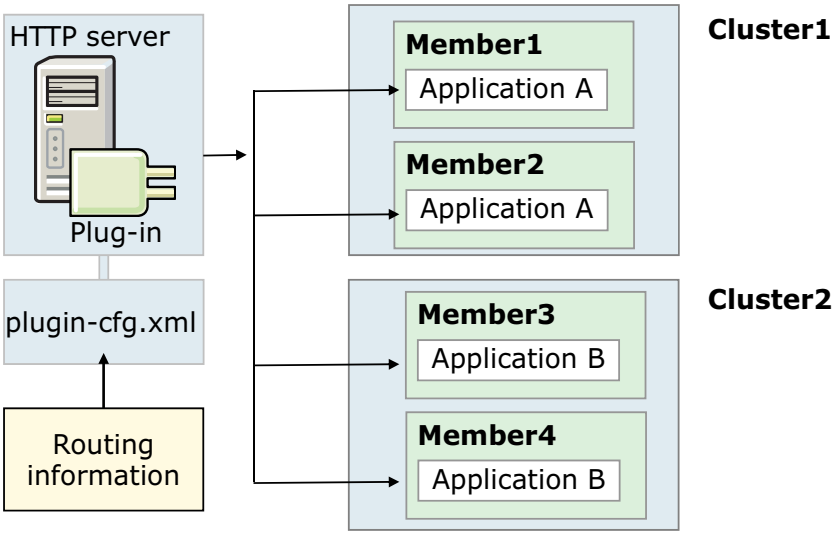






*Figure 4-5. Web server plug-in routing (2 of 2)*

With web server plug-in routing, as the routing topology becomes more complex (servers or applications added for example), the routing information file must be regenerated for each change. The changes must then be merged into the routing information file that is provided to the WebSphere plug-in. The size of the routing configuration file grows as the number of servers and applications grows.

The WebSphere plug-in detects when a server or application is unavailable, 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.

# 4.2. Dynamic Routing overview

IBM Training

# Dynamic Routing overview

Dynamic Routing



*Figure 4-6. Dynamic Routing overview*

IBM Training

## Topics

- Web server routing with the WebSphere plug-in
- Dynamic Routing overview
- Configuring Dynamic Routing




*Figure 4-7. Topics*

IBM Training

## Overview of Dynamic Routing

- Feature of Liberty collectives
- Maintains routing information about applications in a collective
- Dynamically delivers routing information to the web server plug-in for WebSphere Liberty
  - Dynamically and automatically obtains updates to WebSphere Liberty deployments
  - No more regeneration of WebSphere plug-in configuration file
  - Information regarding changes to applications, cluster members, servers, or virtual hosts, is dynamically delivered to the web server plug-in
- Enabled by adding the `dynamicRouting-1.0` feature to the collective controller
  - All the routing information is gathered for the collective members




*Figure 4-8. Overview of Dynamic Routing*

The feature `dynamicRouting-1.0` is added to a Liberty collective controller so that all of the routing information can be gathered for the collective members. 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. The `dynamicRouting-1.0` 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.

The main benefit of Dynamic Routing is that routing information is maintained so that web requests are routed successfully as the routing topology changes. The plug-in does not have to use communication errors to determine whether an application or server is available. Also, the plug-in routing configuration file does not need to be maintained manually. This reduces the chances of error in the environment and saves administrative time.

IBM Training

## Dynamic Routing components

- Collective members
  - Publish member-specific routing information to a collective controller
- Collective controller
  - Maintains routing information that is published by members
  - Provides the Dynamic Routing service for the plug-in to connect to
  - Delivers routing information to the plug-in
- Intelligent Management enabled WebSphere plug-in
  - Gets routing information from the Dynamic Routing service
  - Chooses destinations for requests

Dynamic Routing



*Figure 4-9. Dynamic Routing components*

There are three components that cooperate to provide Dynamic Routing function:

- Collective members
- Collective controller
- Intelligent management enabled WebSphere plug-in

Each collective member publishes its specific routing configuration to a collective controller.

Dynamic Routing enabled collective controllers maintain the routing information that the collective members publish. The controllers provide the Dynamic Routing service where the WebSphere plug-in connects. The controllers deliver routing information to the plug-in as the routing configuration or state changes.

The Intelligent Management enabled WebSphere plug-in gets the routing information from the Dynamic Routing service, and chooses where to send requests.

Collective members publish member-specific routing information to the collective controller. The controller uses the published information to create and maintain the routing information for the collective. The controller delivers the routing information to the plug-in as the routing configuration or state changes. As new controllers with the dynamic routing feature enabled are added to the collective, the new controller information is also delivered to the plug-in. The plug-in can use the new controllers to fail over to when the original controller becomes unavailable. The dynamic

routing service maintains the current routing information for all of the applications in the collective. The Intelligent Management-enabled WebSphere plug-in connects to the dynamic routing service and the service delivers up-to-date routing information to the plug-in. As the servers and applications in the collective change, the new routing information is delivered to the plug-in. The plug-in then routes requests successfully into the changed topology. The dynamic routing service also delivers server and application start and stop events to the plug-in. The main benefit of dynamic routing is that routing information is maintained so that web requests are routed successfully as the routing topology changes. The plug-in does not have to use communication errors to determine whether an application or server is available. Also, the plug-in routing configuration file does not need to be maintained manually. This task reduces the chances of error in the environment and saves administrative time.

IBM Training

## Dynamic Routing topology

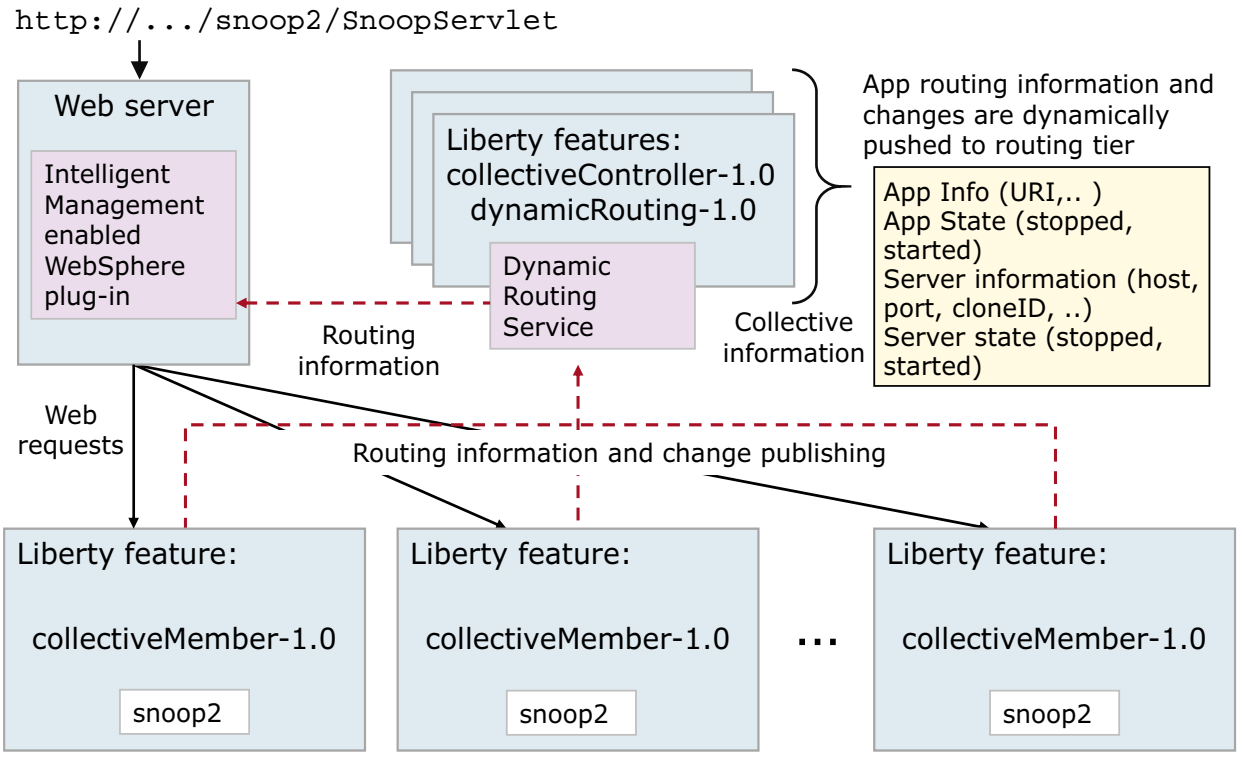






*Figure 4-10. Dynamic Routing topology*

In the example that is shown here, there are three Liberty servers (collective replicas) that can handle requests for the “snoop2” application. The WebSphere plug-in is used by the IBM HTTP Server web server to route requests to a Liberty server.

Collective members publish member-specific routing information to the collective controller.

The controller uses the published information to create and maintain the routing information for the collective. The controller delivers the routing information to the plug-in as the routing configuration or state changes.

As new controllers with the Dynamic Routing feature enabled are added to the collective, the new controller information is also delivered to the plug-in.

The plug-in can use the new controllers to fail over to when the original controller becomes unavailable.

IBM Training

## Benefits of Dynamic Routing

- Application requests are routed correctly as servers and applications in a collective are:
  - Added, removed, or updated
  - Stopped or started
- No need to update WebSphere plug-in configuration file as the collective changes
- Plug-in is informed when servers and applications are started and stopped
- Dynamic Routing enabled controllers are discovered automatically




*Figure 4-11. Benefits of Dynamic Routing*

The main benefit of Dynamic Routing is that routing information is maintained so that web requests are routed successfully as the routing topology changes. As servers and applications are added, removed, or updated, the routing configuration of the Intelligent Management enabled WebSphere plug-in is kept up-to-date. As servers and applications are started and stopped, the new state is delivered to the plug-in. The plug-in reacts more quickly when servers and applications become available or unavailable.

With Dynamic Routing, the plug-in routing configuration file does not need to be maintained manually. Dynamic Routing saves time, and reduces the chance of errors.

Another benefit of Dynamic Routing is that as new controllers with Dynamic Routing enabled are added to the collective, the new controller information is delivered to the WebSphere plug-in.

# 4.3. Configuring Dynamic Routing

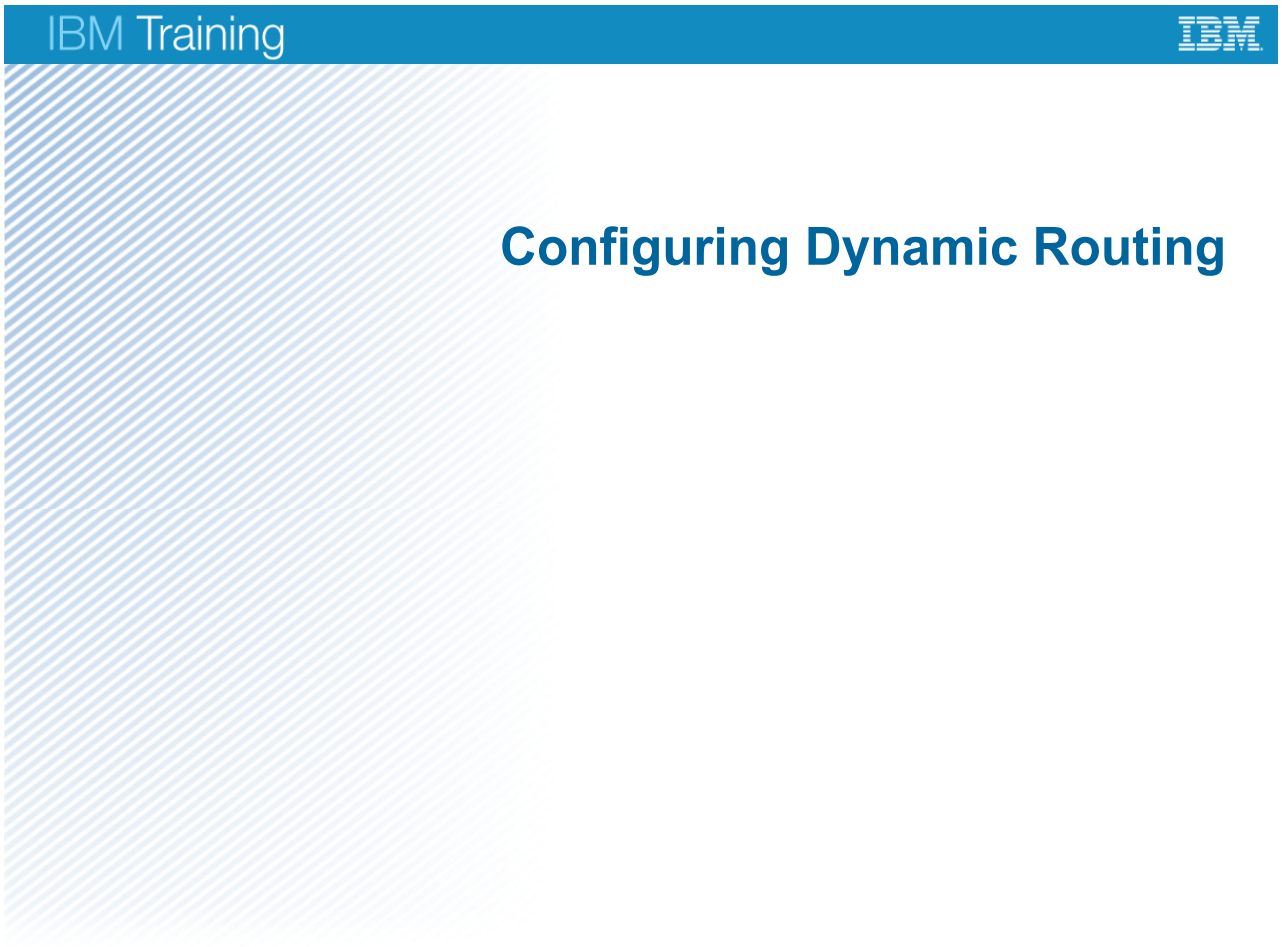



*Figure 4-12. Configuring Dynamic Routing*

IBM Training

## Topics

- Web server routing with the WebSphere plug-in
- Dynamic Routing overview
- Configuring Dynamic Routing




*Figure 4-13. Topics*

## Steps to enable Dynamic Routing

1. Configure a collective
2. Configure a collective controller replica set (optional)
3. Configure clusters (optional)
4. Add the `dynamicRouting-1.0` feature to the `server.xml` file of the controller or controllers
5. Start all the controllers where the Dynamic Routing feature enabled
6. Use the `dynamicRouting` command to set up Dynamic Routing
7. Configure secure communication with the WebSphere plug-in
8. Put the generated Dynamic Routing artifacts where the WebSphere plug-in can read them
9. Start the web server and begin routing to the applications installed in the collective




*Figure 4-14. Steps to enable Dynamic Routing*

To use the dynamic routing feature, you need to configure the `dynanamicRouting-1.0` feature in the `server.xml` file on the controller. The `dynanamicRouting-1.0` feature provides the dynamic routing service.

There are the steps that are completed to configure Dynamic Routing. The first three steps are used to configure Liberty collectives, whether the Dynamic Routing feature is used.

The remaining steps are specifically for configuring Dynamic Routing in a collective. You must add the `dynanamicRouting-1.0` feature to one or more collective controllers. You must use the `dynamicRouting` command to set up Dynamic Routing, and you must configure secure communication with the plug-in. Finally, you must make the Dynamic Routing artifacts available to the WebSphere plug-in.

The steps that deal specifically with Dynamic Routing are discussed in more detail on the following slides.

The first step is to install a web server. The Web Server plug-in for WebSphere Application Server, such as the IBM HTTP Server can be used. Then, download and install the IBM Installation Manager. You can use the Installation Manager to access online product repositories to install the Web Server plug-in for WebSphere Application Server.

IBM Training

## The dynamicRouting command

- `genPluginCfg` command
  - Creates the `plugin-cfg.xml` file that is used by the WebSphere plug-in
  - Enables Intelligent Management by using the `<IntelligentManagement>` stanza
- `genKeystore` command
  - Creates the security artifacts that are needed for communication with the plug-in
  - Generates a keystore that contains a personal certificate and signer certificates
- `setup` command
  - Completes both operations, `genPluginCfg` and `genKeystore`
  - Creates a `plugin-cfg.xml` and `plugin-key.jks` or `plugin-key.p12`

```
dynamicRouting setup --port=9443 --host=wlphost
--user=admin --password=passw0rd --keystorePassword=webAS
--pluginInstallRoot=/opt/IBM/WebSphere/Plugins/
--webServerNames=webserver1
```

Dynamic Routing



*Figure 4-15. The dynamicRouting command*

The `dynamicRouting` command provides two functions. The command generates the plug-in configuration file that is needed by the WebSphere plug-in. The generated file contains the information that is needed to connect to the Dynamic Routing service that is running in one or more collective controllers. The genPluginCfg action generates the routing configuration file.

The `dynamicRouting` command also generates the security artifacts that are needed to establish secure communication between the WebSphere plug-in and the Dynamic Routing service. The `genKeystore` action generates the required security artifacts.

To generate the keystore and plug-in configuration files, you use the `dynamicRouting setup` command. The `--host` and `--port` arguments identify the collective controller that can process the command. The `--user` and `--password` arguments are the administrative user ID and password for authenticating with the controller. If you do not provide the password value on the command line, you are prompted to enter it when running the command. You also need to include details for the `--pluginInstallRoot` and `--webServerNames` arguments.

The output of the command is the plug-in configuration file, `plugin-cfg.xml`, and a keystore which contains a personal and signer certificates, `plugin-key.jks`. Both of the files are generated in the directory from where you ran the `dynamicRouting setup` command.

IBM Training

# Required parameters: dynamicRouting genPluginCfg

| Parameter | Description |
|---|---|
| <first parameter> | genPluginCfg |
| --host | The host name where the Dynamic Routing enabled collective controller is running |
| --port | The https port where the collective controller is listening |
| --user | The user name to use when you connect to the collective controller |
| --password | The password to use when you connect to the collective controller |
| --pluginInstallRoot | Fully qualified path of the WebSphere plug-in root directory on the web server host |
| --webServerNames | Comma-separated names of the web servers for which WebSphere plug-in files must be generated |

Creates a **plugin-cfg.xml** file in the directory where the command is run

Dynamic Routing



*Figure 4-16. Required parameters: dynamicRouting genPluginCfg*

Here are the detail parameters for the `genPluginCfg` command. All options are required.

The `--host` and `--port` options specify the host and port of the controller where the Dynamic Routing feature is configured.

The `--user` and `--password` options specify the credentials for an administrative user in the collective.

The `--pluginInstallRoot` option specifies the path of the WebSphere plug-in root directory on the web server host.

The `--webServerNames` option specifies the names of the web servers for which the plug-in configuration files must be generated.

Creates the file `plugin-cfg.xml` in directory where the command is run.

## Example plugin-cfg.xml file

- One <Connector> stanza is required for each collective controller in the collective with dynamic routing enabled

```
<IntelligentManagement>
    <Property name="webserverName" value="webServer1"/>

    <ConnectorCluster enabled="true" maxRetries="-1"
    name="default“ retryInterval="60">
       <Property name="uri" value="/ibm/api/dynamicRouting"/>
       <Connector host="controller1.acme.com" port="9444"
                  protocol="https">

<Property name="keyring“
value="/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-
key.kdb"/>

       </Connector>
    </ConnectorCluster>
</IntelligentManagement>
```




*Figure 4-17. Example plugin-cfg.xml file*

Here is an example of part of the `plugin-cfg.xml` file that the `genPluginCfg` action of the `dynamicRouting` command generates.

The <IntelligentManagement> stanza is inserted in the file.

The <ConnectorCluster> stanza contains one <Connector> stanza for each Dynamic Routing enabled controller that was in the collective when the command was run.

The details of the `plugin-cfg.xml` file are shown on the slide. Note the element and details in the stanza. Included is information about the connector, which indicates the controller details. If you have multiple controllers, there is a connector element for each controller.

IBM Training

# Parameters: dynamicRouting genKeystore

| Parameter | Description |
|---|---|
| <first parameter> | genKeystore |
| --host | The host name of the target collective controller |
| --port | The https port where the collective controller is listening |
| --user | The user name to use when you connect to the collective controller |
| --password | The password to use when you connect to the collective controller |
| --keystorePassword | The password for the generated keystore |
| --keystoreType | (Optional) The type of the generated keystore. Default type is JKS. Valid values are JKS and PKCS12. |
| --certificateSubject=DN | (Optional) The DN for the generated SSL certificate. The default value, if unspecified is CN=*value of --user argument*,OU-client,O=ibm,C=us. |




*Figure 4-18. Parameters: dynamicRouting genKeystore*

Here are the details of the `genKeystore` action. The host and port options specify the host and port of the controller where the Dynamic Routing feature is configured.

The `--user` and `--password` options specify the credentials for an administrative user in the collective.

The `--keystorePassword` is the password for the generated keystore. If no value is specified, you are prompted.

The last two arguments are optional.

The `--certificateSubject` option is how you specify the Distinguished Name (DN) used in the generated certificate.

The `--keystoreType` option is how you choose between generating a jks type keystore or a pkcs12 type keystore.

To get online help, enter: `<wlp_root>/bin/dynamicRouting help genKeyStore`

IBM Training

## Purpose of the dynamicRouting genKeystore command

- Generates a keystore in JKS (default) or PKCS12 format
  - Use the `--keystoreType` parameter to specify the keystore format
  - Generates a file that is named `plugin-key.jks` or `plugin-key.p12`
  - Copy the generated to a temporary directory on the web server host
- Keystore contains
  - A personal certificate
  - Both the *memberRoot* and *controllerRoot* signed certificates
- The keystore is used to send HTTPS requests to applications in the Liberty collective members




*Figure 4-19. Purpose of the dynamicRouting genKeystore command*

This action generates a keystore in PKCS12 or JKS format. Use the `--keystoreType` parameter to specify the keystore format.

Generation of the PKCS12 keystore is only supported when you use IBM Java. The keystore contains a personal certificate and both the memberRoot and controllerRoot signer certificates that allow secure communication between the WebSphere plug-in and the Dynamic Routing service. The same keystore is also be used to send HTTPS requests to applications in the Liberty collective members.

IBM Training

## Set up secure communication with the WebSphere plug-in

- Convert the keystore to CMS (Certificate Management Services) format
  - Supported format of the WebSphere plug-in
  - Use the `gskcmd` (included with the web server package) to convert the keystore
  - Set the personal certificate as the default

```
gskcmd -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


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

- Generated files:
  - `plugin-key.kdb`, `plugin-key.rdb`, and `plugin-key.sth`




*Figure 4-20. Set up secure communication with the WebSphere plug-in*

Set up secure communication between the plug-in and the dynamic routing service. The keystore that is generated by the `dynamicRouting setup` command, `plugin-key.jks`, is not in a format that the WebSphere plug-in can use. The generated file must be converted to a format that the plug-in can use, the CMS format.

The command to convert the keystore is provided with the web server installation. Use the `gskcmd` to complete the conversion.

This slide shows an example of the commands that are needed to create the keystore in the required format.

---

**Information**

The `gskcapicmd` command provides a command line interface for certificate management tasks that the ikeyman command provides.

The `gskcmd` command is a Java-based alternative.

Native and Java supplemental command-line certificate management tools are also provided in the IBM HTTP Server `bin/` directory as `gskcmd` (also known as `iKeycmd`) and `gskcapicmd` (also

known as `gsk8capicmd`). Both share similar syntax and contain extensive embedded usage information.

## Make the artifacts available to the WebSphere plug-in

- Verify the files are in the `<pluginInstallRoot>/config/<web server name>/` directory
  - The `gskcmd` command creates files in this directory
  - Copy the `plugin-cfg.xml` file to the directory
- Start the web server and begin routing to the application installed in the collective

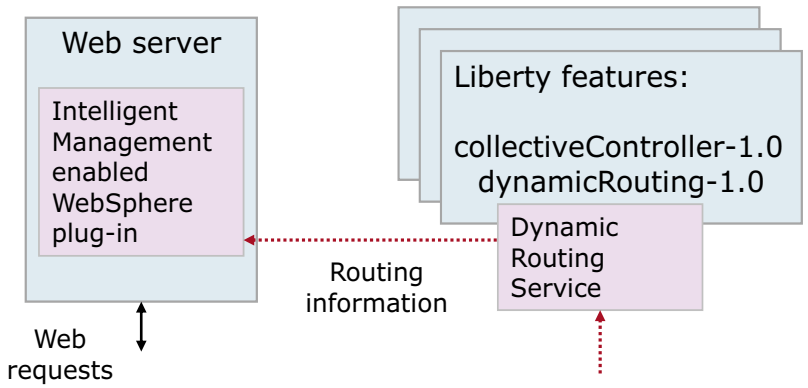






*Figure 4-21. Make the artifacts available to the WebSphere plug-in*

Make the necessary files available to the WebSphere plug-in.

Copy both the `plugin-cfg.xml` file and all of the converted keystore files to the plug-in config directory. Finally, copy the `plugin-cfg.xml` to the directory specified in the `WebSpherePluginConfig` directive in the IBM HTTP Server (IHS) `httpd.conf` file. Then, start the web server and begin dynamically routing to the applications installed in the Liberty collective.

The generated files are copied to the correct directories on the host where the web server is installed.

The CMS formatted store file *(`.kdb`)* must be moved to the directory <value of the `--pluginInstallRoot` argument> `/config/<web server name>/`. The *`.rdb`* and *`.sth`* files must also be moved to the same directory.

Copy the files that the `gskcmd` creates from the temporary directory to the directory `<pluginInstallRoot>/config/<web server name>`.

IBM Training

## Unit summary

- Describe routing by the web server when using the WebSphere plug-in
- Describe the purpose of Dynamic Routing
- Describe benefits of Dynamic Routing
- Describe the components that are used in Dynamic Routing
- Explain how to configure Dynamic Routing




*Figure 4-22. Unit summary*

IBM Training

## Review questions

1. True or False: The `dynamicRouting genKeystore` command generates a keystore that is used by the WebSphere plug-in.

2. The Dynamic Routing is a feature of:
   A. WebSphere plug-in
   B. Clusters
   C. Web server
   D. Collectives

3. True or False: The `<IntelligentManagement>` stanza generated to `plugin-cfg.xml` file.




*Figure 4-23. Review questions*

Write your answers here:

1.

2.

3.

## Review answers

1. True or False: The `dynamicRouting genKeystore` command generates a keystore that is used by the WebSphere plug-in.
   The answer is False.
   The generated keystore must be converted into a CMS keystore (`.kdb`) before it is used by the WebSphere plug-in.

2. The Dynamic Routing is a feature of:
   A. WebSphere plug-in
   B. Clusters
   C. Web server
   D. Collectives
   The answer is D.

3. True or False: The `<IntelligentManagement>` stanza generated to `plugin-cfg.xml` file.
   The answer is True.




*Figure 4-24. Review answers*

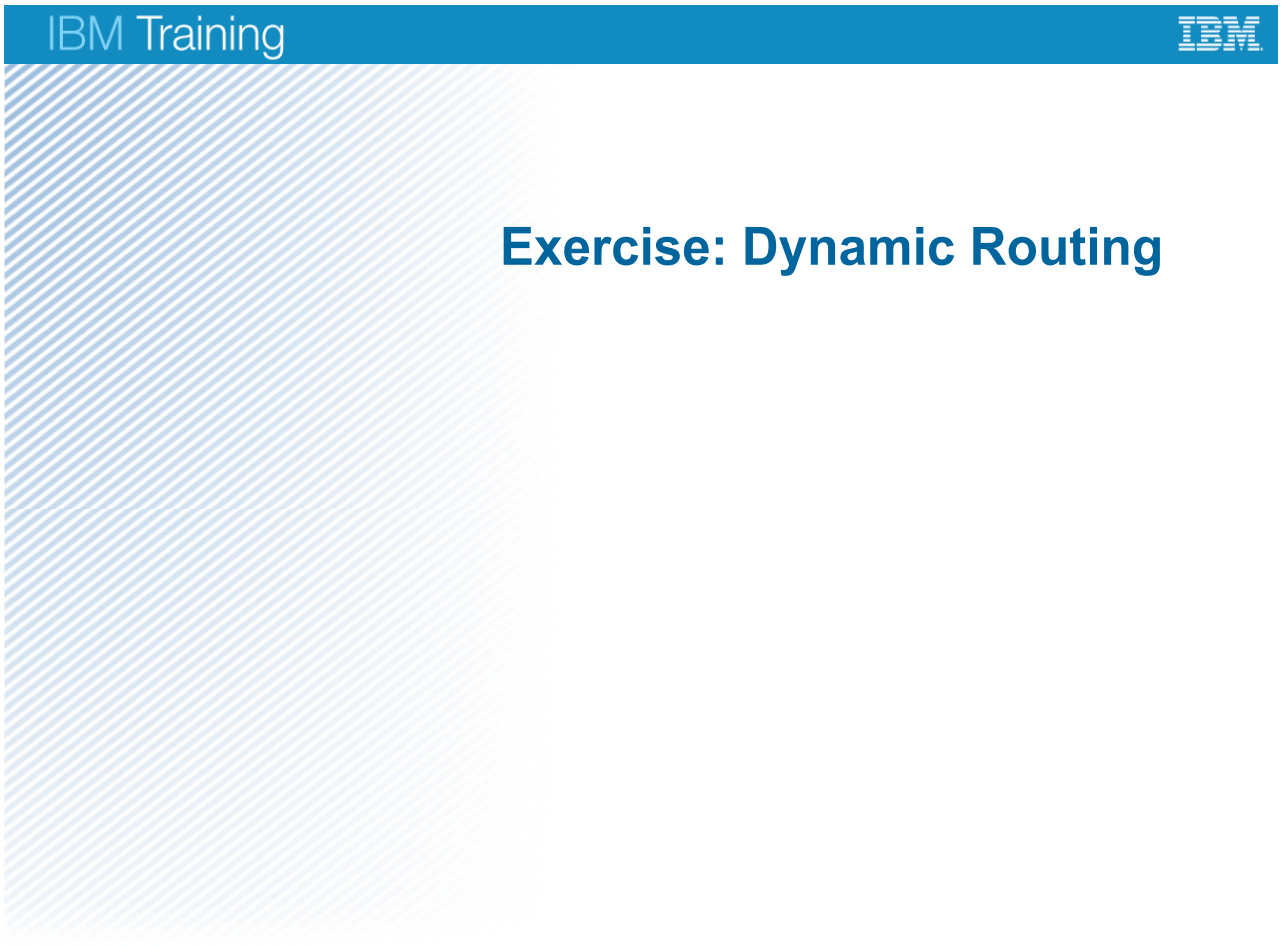



*Figure 4-25. Exercise: Dynamic Routing*

IBM Training

## Exercise objectives

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




*Figure 4-26. Exercise objectives*

# Unit 5. Auto-scaling in Liberty

## Estimated time

00:45

## Overview

In this unit, you learn how auto-scaling in Liberty provides elasticity to a clustered server environment.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

IBM Training

## Unit objectives

- Describe the purpose of auto-scaling
- Describe auto-scaling topology
- Describe the benefits of auto-scaling
- Describe auto-scaling policies
- Describe the components that are used in auto-scaling
- Explain how to configure auto-scaling




*Figure 5-1. Unit objectives*

## What is auto scaling

- Auto scaling dynamically adjusts the number of running Liberty servers in a cluster based on workload
  - Adjust the number of JVMs
  - Adjust the number of Liberty servers
  - An auto scaling cluster is a server cluster that can expand and contract depending on the workload in your environment
- Auto scaling decision making is defined through scaling policies and the current state of the cluster members
  - Controlled by using policies when the consumption of resources rises above an upper threshold or falls below a lower threshold
- Feature does not require the use of Dynamic Routing
  - However, auto scaling performs well when both features are used together
- Feature is only available in WebSphere Application Server Liberty Network Deployment




*Figure 5-2. What is auto scaling*

Auto scaling provides an autonomic scaling capability of Liberty servers. Auto scaling provides JVM elasticity to clusters. With JVM elasticity, auto scaling features dynamically adjust the number of running Liberty servers in a cluster based on workload. As workload goes up, auto scaling starts cluster members. When workload goes down, auto scaling stops cluster members.

With the auto scaling feature enabled, you can now dynamically adjust the number of Java virtual machines (JVMs) or Liberty servers that are used to service your workload.

An auto scaling cluster is a server cluster that can expand and contract which depends on the workload in your environment. Auto scaling capabilities are controlled by using policies when the consumption of resources rises above an upper threshold or falls below a lower threshold.

## Auto scaling components

- Scaling controller
  - Enabled with `scalingController-1.0` feature
  - Keeps the last known state of the scaling members
  - Analyzes incoming resource metrics
  - Decides when to expand or contract an auto scaling cluster
  - High availability uses collective replica sets function
- Scaling member
  - Enabled with `scalingMember-1.0` feature
  - Monitors the workload within the server and its host
  - Each scaling member needs to define a `hostSingleton` element with a port in the `server.xml` file
  - All members on the same host and cluster must use the same port
  - The port is used by an election service to identify a host leader
  - The host leader sends member information back to the scaling controller

```
<hostSingleton name="MemberSingletonService" port="5164" />
```




*Figure 5-3. Auto scaling components*

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

- Scaling controller
- Scaling members

The scaling controller maintains the view of all the clusters in the topology and is in charge of starting and stopping cluster members and provisioning new servers. The scaling controller does this task by analyzing performance metrics that are sent by cluster members and comparing them against defined scaling policies. When a metric violates a policy threshold, it decides whether to scale in or scale out.

Collective controllers are required because they provide administration functions that uses the ability of the collective controller to manage the scaling controller. If replica sets are being used, all collective controller members must be scaling controllers. However, only one of the running scaling controllers can make decisions. When using replica sets if a controller is stopped, another running scaling controller takes over for it.

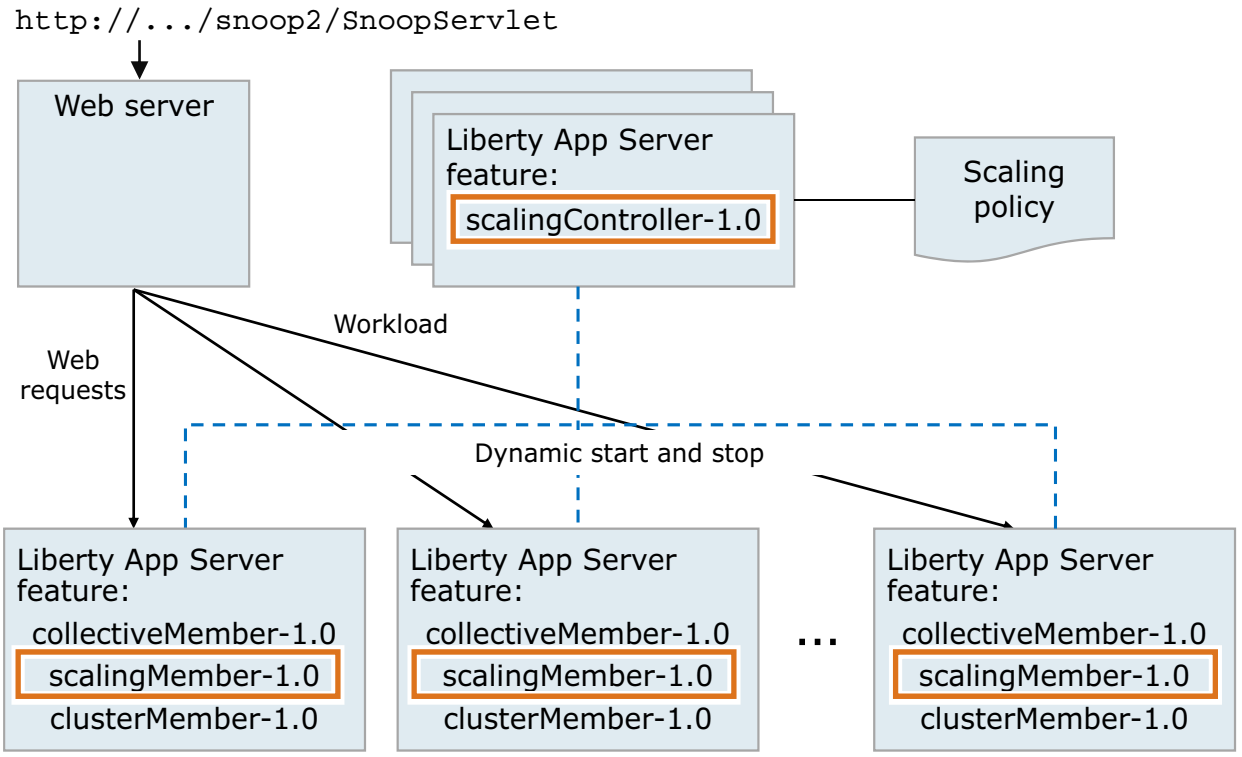
Scaling members are collective members. The scaling member monitors the resources within the server and its host and when needed, it sends this information back to the scaling controller. Scaling members are divided into two categories:

- Host leaders
- Host followers

Host leaders are scaling members that are elected to talk to the scaling controller. Only one scaling member on a host is elected the host leader. All other members of the cluster then become host followers who monitor their resource usage and pass reports to the host leader. The host leader is the only scaling member that communicates with the scaling controller. All scaling member servers must belong to a cluster because all scaling policy information is applied at the cluster level. A cluster has a unique name within a Liberty collective. All Liberty servers that specify the same cluster name within the same collective are members of the same cluster.

IBM Training

## Auto scaling topology




Auto-scaling in Liberty



*Figure 5-4. Auto scaling topology*

Here is the simplest topology of auto scaling for Liberty. An application cluster, and each server in the cluster is a scaling member. Each scaling member has the `scalingMember-1.0` feature and at least one more server has the `scalingController-1.0` feature. The scaling policy defines conditions for when scaling members are started or stopped.

Auto scaling provides autonomic control over all participating clusters and their members.

IBM Training

## Benefits of auto scaling

- Provides elasticity for application clusters
  - Automated monitoring and control of clusters
  - Decreases manual administration
- Centralized policy management
  - Policy is specified at the controller
  - Specify default policy and then customize by cluster

The scaling controller supports two usage scenarios

- JVM elasticity
  - Starts and stops Liberty servers based on resource usage and scaling policies
  - No provisioning of new servers
- Liberty elasticity
  - Can install Liberty servers onto registered hosts and provision new servers
  - Start and stop servers based on usage and scaling policies




*Figure 5-5. Benefits of auto scaling*

One benefit of auto scaling is the elasticity that it provides application clusters. Auto scaling offers a way to automate cluster monitoring, relieving an administrator of the need to gauge how many servers are needed to support the application at any one time.

A default policy is set that applies to all clusters, which can then be extended to support the unique requirements of specific clusters. All policy is stated in the server.xml file of the controller and thus does not have to be stated in each member of a cluster.

The scaling controller supports the following usage scenarios:

- JVM elasticity
  The scaling controller starts and stops existing Liberty servers that are based on resource usage and optional scaling policies. There is no provisioning of new servers.
- Liberty elasticity
  The scaling controller can install Liberty software onto registered hosts and provision new servers, and then start and stop those servers, based on resource usage and optional scaling policies.

With Liberty elasticity, auto scaling features can install Liberty software onto a registered host and create a server. The number of available servers grows when application demand is high and

shrinks when application demand is low. Because support for Liberty elasticity includes support for JVM elasticity, auto scaling can also start or stop servers based on workload.

IBM Training

## Overview of scaling policies

- Scaling policies are used by the scaling controller to determine when to start or stop members of a cluster
- Two ways to scale a cluster
  - Scale to meet the minimum or maximum number of servers per cluster
  - Scale to meet the resource demands of a cluster
- A built-in scaling policy is embedded in the scaling controller
  - Can be overridden by defining default policy or a scaling policy
- Built-in scaling policy definitions:
  - A minimum of two cluster members, if available, are kept active
  - Another cluster member is started when the average CPU, heap, and memory use of all active members exceeds 90%
  - A cluster member is stopped when the average CPU and heap use drops below 30%




*Figure 5-6. Overview of scaling policies*

You can use scaling policies 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.

To dynamically adjust the number of servers that are used to service your workload, auto scaling uses user-defined scaling policies with the current state of the cluster to determine whether a scaling action needs to occur. The machines that host the cluster members provide the data on the workload.

Scaling policies are used by the scaling controller to determine when to start or stop members of a cluster.

Two ways to scale a cluster:

- Scale to meet the minimum or maximum number of servers per cluster.
- Scale to meet the resource demands of a cluster.

All scaling policies must be placed in the `server.xml` of the scaling controller.

Scaling can also happen based on the resource consumption of a cluster. You can use scaling policies to set the minimum and maximum thresholds for server resources, such as CPU, memory,

and heap. The scaling member sends the metric resource data to the scaling controller. If a cluster violates a metric that is defined in a scaling policy, the scaling controller reacts, starting a member if a maximum is breached, or stopping a member if usage drops below a minimum.

By default, a built-in scaling policy is embedded in the scaling controller.

The built-in scaling policy indicates:

- A minimum of two cluster members, if available, are kept active. The minimum number might not be met if some or all of the members are exceeding the metric thresholds.
- An extra cluster member is started when the average CPU, heap, or memory use of all active members exceeds 90%.
- A cluster member is stopped when the average CPU and heap use drops below 30%.

## Modifying scaling policies

- Default scaling policy
  - Define a single default scaling policy to *manage all clusters* that do not need a more specific scaling policy defined
  - Example: Set the minimum number of active cluster members to 3

```
<scalingDefinitions>
  <defaultScalingPolicy min="3" />
</scalingDefinitions>
```

- Scaling policy
  - Define a scaling policy to manage one or more clusters with targeted criteria
  - Example: Changes the CPU thresholds for stopping and starting servers in a cluster named dynamicCluster

```
<scalingDefinitions>
     <scalingPolicy id="cluster1Policy" min="2" max="5">
          <bind clusters="dynamicCluster"/>
          <metric name="CPU" min="10" max="70"/>
     </scalingPolicy>
</scalingDefinitions>
```




*Figure 5-7. Modifying scaling policies*

Scaling policies are defined in the `server.xml` file of the scaling controller with the `scalingDefinitions` element.

You can define a single default scaling policy to manage all clusters that do not need a more specific scaling policy defined. The `defaultScalingPolicy` stanza defines the default scaling policy. The default scaling policy includes built-in policy metrics, including min and max values. Any changes that are made to the default scaling policy overrides the built-in values.

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 element. The `defaultScalingPolicy` indicates the default policy to use for any auto-scaled cluster that does not have a specific scaling policy defined.

Define a scaling policy to manage one or more clusters with targeted criteria.

The example scaling policy on the slide is defined for the cluster named “dynamicCluster.” The policy part of this definition is the metric, which defines that the CPU percent that is used should be kept between 10% and 70%.

You can also override the built-in policy by using the element. The element provides fine-grained control of each cluster’s scaling policy. The thresholds that are defined in the policy are targeted at a specific cluster or clusters by using the element. In addition, the metric policy for an element must

be defined, or they will not exist. This allows granular control of which metrics are analyzed when making scaling decisions.

## Scaling policies min and max values

- You can use scaling policies to set a minimum and maximum number of scaling members for a cluster
- The controller uses this information to ensure that the number of running servers falls within this value
- Example: If min=2 and only one scaling member is started, the controller starts another member
- Example: If min=3 and four members are started, the controller stops one member




*Figure 5-8. Scaling policies min and max values*

One way to scale is by enforcing a minimum and maximum number of servers per cluster. If the controller monitoring a cluster notices that the number of started instances is below the minimum, it starts stopped-servers. Similarly, if it notices that too many instances are started, it stops servers until the cluster falls within its bounds.

## Disable auto scaling

- The Liberty configuration is dynamic
  - When you add the scaling controller feature, the default scaling policy takes effect
- Disable the default scaling policy while you configure auto scaling
  - Set the default scaling policy parameter enabled="false"

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

<scalingDefinitions>
  <defaultScalingPolicy enabled="false" min="2" max="4" />
</scalingDefinitions>
```




*Figure 5-9. Disable auto scaling*

Because Liberty configuration is dynamic, when you add the scaling controller feature, the default scaling policy takes effect and you might get unexpected results. For example, the default policy has a minimum of two servers. When you save the `server.xml` file, the scaling controller attempts to start two servers. For this reason, it is best to disable the default scaling policy while configuring auto scaling.

Setting the enabled value to false prevents the scaling controller from making scaling decisions.

## Scaling definition example

- The default scaling policy in this example shows minimum and maximum number of servers and thresholds for the cpu, heap, and memory metrics
- One server is stopped (scaled in) when the average of the cpu and heap drops below the minimum threshold
- One server is started (scaled out) when the average of the all the metrics exceeds the maximum threshold

```
<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 units="instance" amount="1" mininterval="30s"/>
  </defaultScalingPolicy>
</scalingDefinitions>
```




*Figure 5-10. Scaling definition example*

The example shows a default scaling policy definition that includes the resource demands metrics and the scaling in and out attributes.

The policy requires at least one member be running and no more than four should be running.

IBM Training

## Scaling definitions in or out

- Scaling `in` defines detailed controls for reducing the number of servers
- Scaling `out` defines detailed controls for increasing the number of servers

| Attribute | Data type | Default value | Description |
|---|---|---|---|
| amount | int<br>Minimum: 1 | 1 | The amount which to scale in units defined by the units attribute |
| minInterval | A period of time in ms | 5 ms | The minimum amount of time between a scaling change |
| units | percentage or instance | instance | The units by which to scale<br>**percentage**: Amount is a percentage relative to the current number of instances<br>**instance**: Amount is a number of instances |




*Figure 5-11. Scaling definitions in or out*

In scaling policy, scaling in refers to reducing the number of servers, while scaling out refers to increasing the number of servers in a cluster.

The table shows the attributes that are used when scaling in or out.

These attributes are specified at the default scaling policy level.

*Scale out* decisions are made on an individual metric basis. All monitored metrics are analyzed for the cluster and if any one of the metrics exceeds the policy max threshold, a scale out event is triggered. *Scale in*decisions are made based on all monitored metrics. The cluster average for each monitored metric is analyzed. If all metrics are below their policy min threshold, a scale in event is triggered.

After a scaling decision is made, the scaling controller selects a scaling target. The scaling target is the host on which the server stop (for scale in) or server start (for scale out) action is run. When determining a target for a scale out action, the host level metrics are considered. If any of the host level metrics exceed that metric's policy max threshold, the scaling controller avoids that host and chooses another host for the scale out action.

The scaling policy can influence the selection of the scaling target. If the policy specifies `scalingPreference="horizontal"`, the scaling controller starts a server on the eligible host with the fewest active servers and stops a server on the host with the most active

servers. If the policy `specifies scalingPreference="vertical"`, the scaling controller starts a server on the eligible host with the most active servers and stops a server on the host with the fewest active servers. The scaling controller attempts to place servers in the same cluster on different hosts when possible, regardless of which scaling preference is chosen.

## Auto scaling clusters for JVM elasticity

- Can configure a collective to support Java virtual machine (JVM) elasticity
- The scaling controller can start or stop Liberty servers based on resource use and scaling policies
  - Only the servers that are already in the collective are eligible for scaling
  - No provisioning of new servers
  - If a maximum resource threshold is reached and there are no additional servers available to scale to, a message is written to the `messages.log` file of the controller

```
[ ] 00000032 com.ibm.ws.imf.apc.IMContainerGroupImpl I
CWWKV0405I: The scaling controller cannot meet the
maximum instances for cluster dynamicCluster because too
few scaling members are defined.
[ ] 00000138 com.ibm.ws.imf.apc.IMContainerGroupImpl I
CWWKV0404I: The scaling controller cannot increase the
number of servers in cluster dynamicCluster because it
cannot find a host with the capacity to add a server.
```




*Figure 5-12. Auto scaling clusters for JVM elasticity*

Collective support for Java virtual machine (JVM) elasticity means that the scaling controller can start or stop Liberty servers that are already in a collective. There is no provisioning of new servers.

JVM elasticity is vertical scalability where servers are started or stopped on the same host.

After adding the scaling controller feature, examine the `messages.log` file to ensure that the feature is installed and activated. Look for the message that indicates the server is selected to be the primary scaling controller. You can also see messages regarding the controller examining the scaling policies.

## JVM elasticity

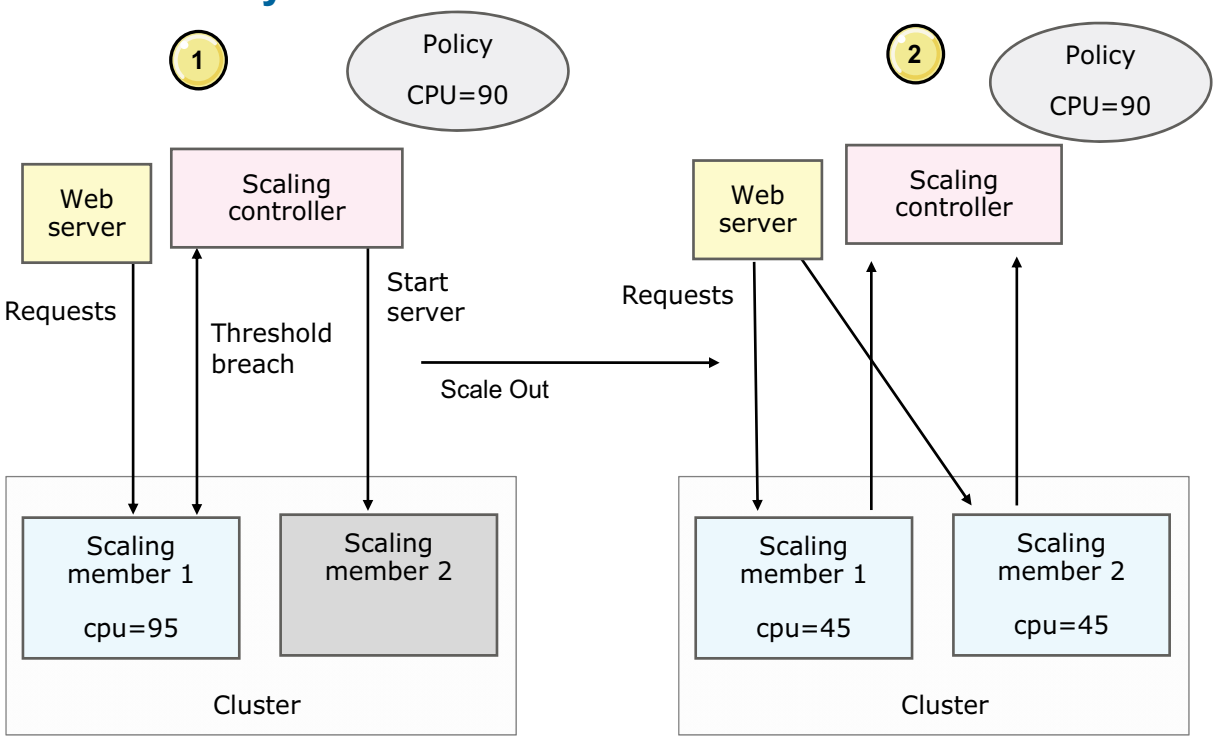






*Figure 5-13. JVM elasticity*

The graphic on the slide shows an example of a scaling policy in use in Java virtual machine (JVM) elasticity. In this example, a cluster member processes requests from a web server. Eventually, the traffic becomes heavy enough to cause it to violate the CPU threshold, triggering a message containing the metrics data to be sent to the scaling controller. The scaling controller, noticing that the member violated the upper threshold, starts another cluster member to balance the load of requests. With another member started, the CPU of both members stabilizes to something reasonable.

## Steps to config auto-scalable clusters for JVM elasticity

1. Configure a collective
2. Add the `scalingController-1.0` feature to the server.xml file of the controller/s
3. Change the default values of the scaling policies to meet the needs of your environment (optional)
4. Add the `scalingMember-1.0` features to the server.xml file of all collective members that you want to participate in auto scaling
5. Add the `hostSingleton` element with a port number to the server.xml file of all collective members that you want to participate in auto scaling
6. Start the controller and the cluster members
7. Start the web server and begin routing to the applications installed in the collective




*Figure 5-14. Steps to config auto-scalable clusters for JVM elasticity*

When you configure JVM elasticity, the scaling controller can start or stop Liberty servers based on resource use and scaling policies. There is no provisioning of new servers.

IBM Training

## Auto scaling clusters for Liberty elasticity

- Scaling controller can install Liberty software onto registered hosts and provision new servers
  - Each target host needs to support the transfer of files to the host and a Java Runtime Environment (JRE) installed
  - Register each host with the scaling controller
- Create and configure installable files and packages that a scaling controller can deploy onto a registered host
- Minimally, you want the scaling controller to install the following installable and package files on a target host:
  - Package with a Liberty server and application
  - Liberty runtime




*Figure 5-15. Auto scaling clusters for Liberty elasticity*

Auto scaling clusters for Liberty elasticity involves horizontal scaling where new servers are deployed and started on other hosts.

To enable Liberty elasticity, each target host needs Remote Execution and Access (RXA) or Secure Shell (SSH) and a Java Runtime Environment (JRE) installed that meets the requirements of the Liberty server.

Registering a host enables the scaling controller to transfer files to that host, and access files, commands, and other resources on the host.

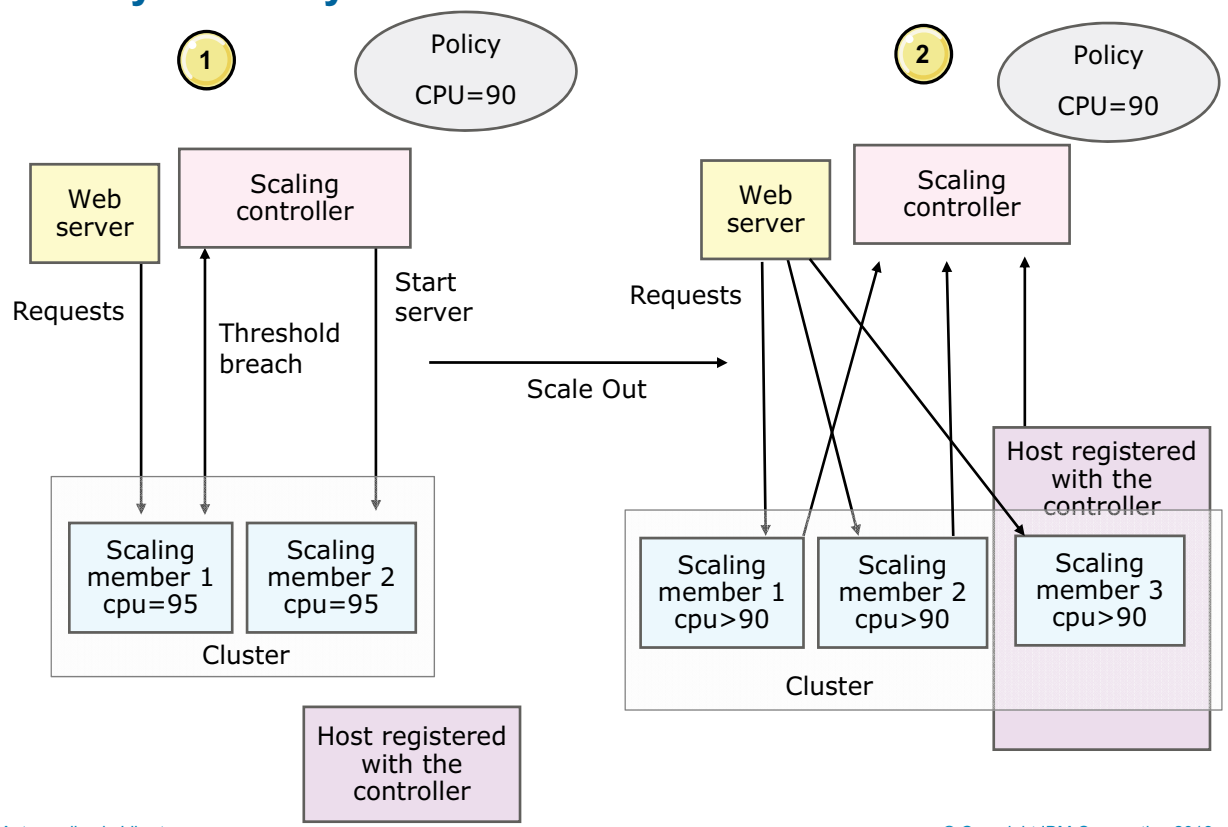
Use the `registerHost` command to register a target host.

Create and configure installables and packages that a scaling controller can deploy onto a registered host.

An *installable* is a binary file that you want to install on a registered host needs to run, such as a Liberty runtime or JRE. A *package* is that application packaged into a compressed file.

IBM Training

## Liberty elasticity




Auto-scaling in Liberty



*Figure 5-16. Liberty elasticity*

The graphic on the slide shows an example of a policy in use in Liberty elasticity. In this example, two cluster members are processing requests from a web server. Again, the traffic becomes heavy enough to cause a violation in the CPU threshold, triggering a message containing the metrics data to be sent to the scaling controller. The scaling controller, noticing that the members violated the upper threshold, starts another cluster member to balance the load of requests. With another member started, the CPU of all members stabilizes to something reasonable.

IBM Training

## Steps to config auto-scalable clusters for Liberty elasticity

1. Complete all of the steps to configure auto-scalable clusters for JVM elasticity
2. Create packages for deploying to new hosts, which includes Liberty runtime and server and JRE
3. Provide packages on the collective controller
4. Register available hosts with collective controller
5. Change the default values of the scaling policies to meet the needs of your environment (optional)
6. Start the controller and the cluster members
7. Start the web server and begin routing to the applications installed in the collective




*Figure 5-17. Steps to config auto-scalable clusters for Liberty elasticity*

When you configure Liberty elasticity, the scaling controller can provision new servers and start or stop Liberty servers based on resource use and scaling policies. When provisioning a new server, the new server is auto-deployed on available hosts that are registered with the collective controller.

With Liberty elasticity, it can deploy the Liberty runtime, Java runtime environment (JRE), and Liberty server packages.

You can configure a collective to support Liberty elasticity. With Liberty elasticity, the scaling controller can install Liberty software onto a registered host and create a server. Also, because support for Liberty elasticity includes support for JVM elasticity, the scaling controller can start or stop Liberty servers based on resource use and optional scaling policies. The number of available servers grows when application demand is high and shrinks when application demand is low.

Determine the target hosts onto which you want to install Liberty software and the Liberty software to install. Minimally, you want the scaling controller to install the following installable and package files on a target host:

A package that provides a stand-alone Liberty server with one application.

An installable that provides a Liberty server that contains the `wlp` directory, but does not contain the `usr` directory.

Each target host needs RXA or SSL and a Java Runtime Environment (JRE) installed that meets the minimum requirements of the Liberty server. If a target host does not have a JRE that is installed with its JAVA_HOME variable and System PATH variable providing a path to the JRE, the scaling controller can install the JRE on the target host.

IBM Training

## Auto scaling with dynamic routing

- Auto scaling in Liberty collectives provides elasticity to a clustered server environment
  - Scaling based on minimum and maximum number of servers
  - Scaling based on resource usage and policy thresholds
- Support for centralized control over auto scaling policy is provided
  - Default scaling policy applies to all clusters
  - More granular scaling policies can be applied to different clusters
- High availability capability is offered with replica sets
- Dynamic Routing uses the auto-scaled resources as they become available
- Dynamic Routing stops using resources that are scaled-in




*Figure 5-18. Auto scaling with dynamic routing*

Auto scaling in Liberty collectives provides elasticity to a clustered server environment.

An auto scaling cluster expands and contracts depending on the workload in your environment.

Dynamic Routing is beneficial when you use the auto-scaling feature of Liberty collectives. The auto-scaling feature starts and stops servers as needed to meet scaling policies. As the server state changes, Dynamic Routing ensures that requests are sent correctly to the available servers without requiring an administrator to update the routing configuration file.

Auto scaling is a configurable feature that runs on top of the collective feature. When used in coordination with the dynamic routing feature, auto scaling can provide elasticity to clusters to support fluctuations in workload demands.

IBM Training

## Unit summary

- Describe the purpose of auto-scaling
- Describe auto-scaling topology
- Describe the benefits of auto-scaling
- Describe auto-scaling policies
- Describe the components that are used in auto-scaling
- Explain how to configure auto-scaling




*Figure 5-19. Unit summary*

## Review questions

1. True or False: JVM elasticity provisions a new server when a policy threshold is breached.

2. True or False: If a cluster member is stopped, you must start the server manually the first time after you add the `scalingMember-1.0` feature to the `server.xml` file.

3. True or False: The `hostSingleton` element in the `server.xml` file of the scaling member defines a port number that is used by an election service to identify a host leader.




*Figure 5-20. Review questions*

Write your answers here:

1.

2.

3.

## Review answers

1. True or False: JVM elasticity provisions a new server when a policy threshold is breached.
   The answer is False.
   JVM elasticity does not provision a new server. Liberty elasticity provisions a new server when a policy threshold is breached.

2. True or False: If a cluster member is stopped, you must start the server manually the first time after you add the `scalingMember-1.0` feature to the `server.xml` file.
   The answer is True.

3. True or False: The `hostSingleton` element in the `server.xml` file of the scaling member defines a port number that is used by an election service to identify a host leader.
   The answer is True.




*Figure 5-21. Review answers*

IBM Training

# Exercise: Auto-scaling




*Figure 5-22. Exercise: Auto-scaling*

## Exercise objectives

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




*Figure 5-23. Exercise objectives*

# Unit 6. Securing Liberty

## Estimated time

00:30

## Overview

In this unit, you learn how to configure security for the Liberty server.

## How you will check your progress

- Review questions
- Lab exercise

## References

IBM Knowledge Center for WebSphere Application Server Liberty Core:
http://www.ibm.com/support/knowledgecenter/SSD28V

IBM Knowledge Center for WebSphere Application Server Liberty Network Deployment:
http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html

IBM Training

# Unit objectives

- Describe basic Liberty security
- Describe authorization in Liberty
- Describe how to configure user registries
- Describe LTPA and SSO in Liberty
- Describe how to enable SSL communication in Liberty




*Figure 6-1. Unit objectives*

IBM Training

## Liberty security basics

- Liberty provides security for the runtime environment and applications
  - User registries
  - Authentication
  - Authorization
- Various Liberty features are applicable to security
  - `appSecurity-2.0`: Security for web applications
  - `ssl-1.0`: Enables SSL
  - `restConnector-1.0`: For remote access by JMX client through a REST-based connector
  - `samlWeb-2.0`: Enables web applications to delegate user authentication to a SAML provider
  - `oauth-2.0`: Authorization to resources by using OAuth2.0 protocol
  - `openid-2.0`: Authentication to multiple entities
  - `passwordUtilities-1.0`: Support for obtaining AuthData from an application
  - `ldapRegistry-3.0`: Support for the LDAP user registry
  - `federatedRegistry-1.0`: Support for federated LDAP user registries

Securing Liberty 

*Figure 6-2. Liberty security basics*

Security is an essential component of any enterprise-level application. Liberty provides support for securing the server runtime environment and applications by using user registries, authentication, and authorization. For secure communication between the client and the server, you can enable SSL for Liberty. A minimal or detailed configuration can be done by adding the `ssl-1.0` server feature to the server configuration file.

When validating the authentication data of a user, the login modules call the user registry that is configured to validate the user information. Liberty supports both a simple configuration-based user registry and a more robust LDAP-based registry.

Authentication confirms the identity of a user. The most common form of authentication is user name and password, such as through basic authentication or form login for web applications. When a user is authenticated, the source of a request is represented as a Subject object at run time.

Authorization determines whether a user has access to a specific role within the system. The Java EE model uses subjects, roles, and role mappings to determine whether access is allowed.

A role is defined within the Java EE application. Some roles, such as the Administrator role, the system predefines. Other roles, the application developer defines. In Java EE, subjects are granted or denied access to a role based on the roles they perform within the application.

## Security basics

- Security by default
  - All opened ports are local host only
  - No remote management by default

*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" />

        <!-- Automatically expand WAR files and EAR files -->
        <applicationManager autoExpand="true"/>
</server>
```




*Figure 6-3. Security basics*

With Liberty Profile, you get security by default. There are no remotely accessible ports.

To access a server from a remote client, the administrator must add the host element to the `httpEndpoint` stanza.

You can see the `server.xml` file on the slide, which shows the `httpEndpoint` information.

## Getting started with security (1 of 3)

- Include the `appSecurity-2.0` and `servlet-3.0` features

```
<featureManager>
     <feature>appSecurity-2.0</feature>
     <feature>servlet-3.0</feature>
</featureManager>
```

- Enable admin security for the Administrator role

```
<quickStartSecurity userName="admin"
     userPassword="adminpwd"/>
```




*Figure 6-4. Getting started with security (1 of 3)*

You can set up a secured Liberty server and web application by following some basic configuration steps. Configuration actions within Liberty are dynamic, which means the configuration updates take effect without having to restart the server.

Create and start your server. Then, include the `appSecurity-2.0` and `servlet-3.0` features in the `server.xml` file. Define the user name and password that is to be granted the Administrator role for server management activities. You can use the `<quickStartSecurity>` element to enable a simple (one user) security setup for Liberty.

## Getting started with security (2 of 3)

- Configure the deployment descriptor with security constraints

```
<!--  SECURITY ROLES  -->
<security-role>
     <role-name>testers</role-name>
<security-role>


<!--  SECURITY CONSTRAINTS  -->
<security-constraint>
     <web-resource-collection>
          <url-pattern>/*</url-pattern>
     </web-resource-collection
     <auth-constaint>
          <role-name>testers</role-name>
     </auth-constraint>
</security-constraint>
```




*Figure 6-5. Getting started with security (2 of 3)*

Configure the deployment descriptor with relevant security constraints to protect web resource. For example, use `<auth-constraint>` and `<role-name>` elements to define a role that can access web resource.

The enforcement of security constraints to applications, and to web service and JMS transports, can be added incrementally to the server configuration by using distinct features. This approach aids the application developer in completing simple unit testing of the application function first, then enabling security for a second, iterative phase of testing.

The slide example `web.xml` file shows that access to all the URIs in the application is protected by the testers role.

# Getting started with security (3 of 3)

- Enable application security

```
<application type="war" id="snoopApp" name="snoopApp"
location="${server.config.dir}/apps/snoopApp.war">
     <application-bnd>
          <security-role name="testers">
               <user name="Ted" />
          </security-role>
     </application-bnd>
<application>
```




*Figure 6-6. Getting started with security (3 of 3)*

To configure authorizations for an application, you can add authorization tables to the application. The server then reads the deployment descriptor of the application to determine whether the user or group has the privilege to access the resource.

Configure your application in the `server.xml` file. In the slide example, the user Ted is mapped to the testers role of the application.

Access your application and log in with the user name Ted. The default URL for the snoopApp application is `http://localhost:9080/snoopApp`.

IBM Training

## Authorization in Liberty

- Include the `appSecurity-2.0` feature
- Configure the authorization information
  - The two types of special-subject are EVERYONE and ALL_AUTHENTICATED_USERS

```
<application type="war" id="myapp" name="myapp"
location="${server.config.dir}/apps/myapp.war">
     <application-bnd>
          <security-role name="user"> <group name="students" />
          </security-role>
          <security-role name="admin"> <user name="jones" />
               <group name="administrators" />
          </security-role>
          <security-role name="AllAuthenticated">
               <special-subject type="ALL_AUTHENTICATED_USERS" />
          </security-role>
     </application-bnd>
</application>
```

Securing Liberty



*Figure 6-7. Authorization in Liberty*

In this example, the admin role is mapped to the user ID jones and all users in the group administrators. The AllAuthenticatedRole is mapped to the special subject ALL_AUTHENTICATED_USERS, meaning that any user has access when they provide valid credentials for authentication.

When you map entities to roles, you can map a special subject instead of a specific user or group. A special subject is an extension to the concept of a subject. A special subject can represent a group of users that fall under a specific category.

The following two types of special subjects are available:

- EVERYONE: This subject represents any entity on the system, which means that no security is available because everyone is allowed access and you are not prompted to enter credentials.
- ALL_AUTHENTICATED_USERS: This subject represents any entity that successfully authenticates to the server.

If the required role is mapped to the EVERYONE special subject, then the authorization service returns immediately to allow anyone access. If the role is mapped to the ALL_AUTHENTICATED_USERS special subject and the user is authenticated, then the authorization service grants access to the user. If none of these conditions are met, then the authorization service determines what users and groups are mapped to the required role. The

authorization service grants access to the resource if the user is mapped to the required role or if the user is part of a group that is mapped to the role.

IBM Training

## OAuth in Liberty

- OAuth is an open standard for delegated authorization
- A user can grant a third-party application access to their information stored with another HTTP service without sharing their access permissions or the full extent of their data
- Authorization to resources by using the `oauth-2.0` feature
  - OAuth provider details are contained in the `server.xml` file
- For complete details, see the following article
  - Using OAuth: Enabling OAuth in the WebSphere Application Server Liberty Profile at http://www.ibm.com/developerworks/websphere/techjournal/1305_odonnell2/1305_odonnell2.html

Securing Liberty



*Figure 6-8. OAuth in Liberty*

OAuth 2.0 service provider support was added to the IBM WebSphere Application Server Liberty profile. OAuth is the de facto standard for delegated authorization across web applications. In addition to delegated access, OAuth is increasingly being used in traditional authentication and authorization roles, driven by the pervasive trends of cloud and mobile. With the OAuth 2.0 specification that is finalized in 2012, companies are quick to adopt the new protocol.

To support the OAuth 2.0 specification, OAuth 2.0 service provider capability was added to the Liberty profile.

A brief overview of the OAuth implementation in the WebSphere Application Server Liberty profile to help understand how the configuration steps affect WebSphere Application Server. This information is also helpful in debugging and tuning the configuration settings. Although users familiar with OAuth in the full profile will likely find OAuth in the Liberty profile easy to configure and use. The OAuth capability in the Liberty profile has several important differences from the full profile.

The Liberty profile provides OAuth service provider support as a single run time security service. OAuth configuration is defined in the server configuration file, and this is the only piece that is required to enable OAuth in the Liberty profile. After an OAuth server is defined in the server configuration file, the server activates the pieces that are needed for OAuth support, such as endpoint servlets and the authorization module. Additionally, with the Liberty server's dynamic

update capability, run time changes can be made directly to the server configuration file, eliminating the need for multiple configuration interfaces.

While the WebSphere Application Server Liberty profile provides all OAuth capabilities needed to protect hosted applications, it does not cover the full spectrum of possible OAuth configurations.

## Liberty administrative security

- One administrator role
- One user registry for applications and administrators
- Simple configuration for a single administrator user

```
<quickStartSecurity userName="admin"
                    userPassword="passw0rd" />

<keystore id="DefaultKeyStore"
              password="passw0rd=" />
```

- Easy configuration for multiple users

```
<administrator-role>
    <user>Ted</user>
    <group>administratorsGroup</group>
</administrator-role>
```




*Figure 6-9. Liberty administrative security*

The administrator role has authorization access to the graphical Admin Center and to run scripts.

More users can be added to the administrators group.

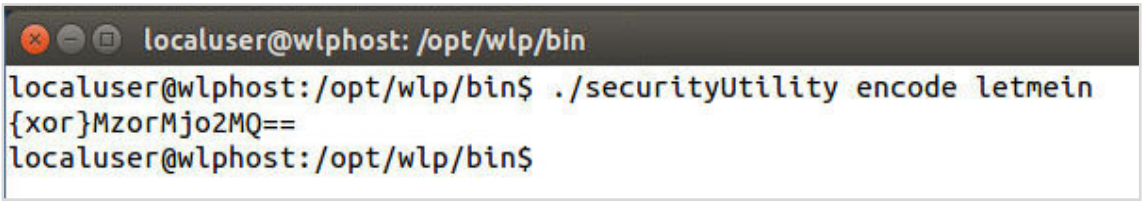
There is only one administrator role for the Liberty profile servers, and a single user registry for both application security and administrative security.

All the JMX methods and MBeans accessed through the REST connector are currently protected with a single role named "administrator". To get started quickly, use the `quickStartSecurity` element to configure a single user with administrator role and configure the default SSL configuration. If you require only a single administrative user, you can use the `quickStartSecurity` configuration element. The required attributes are a user name and password.

For authorizing multiple users of administrative functions, use the administratorRole element in the `server.xml` file to map the users to the administrator role.

## Encoding passwords

- Plain text encoding for password
  - Use the `securityUtility` command
  - Copy the encoded password in the `server.xml` file for users




- WebSphere Application Server Developer Tools for Eclipse encodes the password automatically




*Figure 6-10. Encoding passwords*

Liberty provides a utility to encrypt passwords that are stored in the `server.xml` file. Three mechanisms are supported: Exclusive decisions and merges (XOR) (the default), Advanced Encryption Standard (AES), and hash. It is important to note that encrypting passwords in the configuration files is a common corporate requirement, but does not, in isolation, make the passwords secure. Either the passwords themselves or the encryption key, must be kept in a separate (included) configuration file that is protected by operating system file permissions or some similar secure mechanism.

If you edit the `server.xml` file directly, you can use the `securityUtility encode` command to encode the password for each user. The securityUtility command line tool is available in the `{$INSTALL_ROOT}/bin` directory.

When you run the `securityUtility encode` command, you either supply the password to encode as an input from the command line or, if no arguments are specified, the tool prompts you for the password. The tool then outputs the encoded value. Copy the value output by the tool, and use that value for the password.

Liberty supports Advanced Encryption Standard (AES) encryption for passwords that are stored in the `server.xml` file. When you use this option for protecting system passwords in the Liberty configuration, you need to understand the limits to the protection it provides.

Encrypting a password in the Liberty configuration does not guarantee that the password is secure or protected; it only means that someone who can see the encrypted password, but does not know the encryption key, cannot easily recover the password. The application server process requires access to both the encrypted password and the decryption key, so both these data items need to be stored on the file system that is accessible to the server runtime environment. The encryption key is also required by anyone who encrypts a password that is placed in the server configuration. For an attacker that has access to the same set of files as the Liberty server instance, applying AES encryption to the password therefore provides no additional security over and above "exclusive or" (XOR) encoding.

Nonetheless, there are still reasons why you might consider encrypting passwords in the Liberty configuration. The Liberty configuration is highly composable and sharable. The administration subsystem of traditional (the administrative console and wsadmin scripting) prevents an administrator from gaining access to an XOR-encoded password. Liberty is configured without an administration subsystem, and so any XOR-encoded password is visible to any administrator. Given these design features, consider the following scenarios:

The passwords are not sensitive, so encoding them provides little value.

The passwords are sensitive, so either the configuration files that contain the password are security sensitive and access needs to be controlled, or the passwords are encrypted and the encoding key is then protected as security sensitive.

The encryption key that is used for decrypting can be overridden from the default by setting the `wlp.password.encryption.key` property. This property should not be set in the `server.xml` file that stores the password, but in a separate configuration file that is included by the `server.xml` file. This separate configuration file should contain only a single property declaration, and should be stored outside the normal configuration directory for the server. This ensures that the file that contains the key is not included when you are running the server dump or package command. The encryption key property can also be specified as a bootstrap property. If you choose this option, you should put the encryption key in a separate properties file that is included in the server `bootstrap.properties` file.

IBM Training

## User registries

- Types of registries for authentication
  - Basic registry
  - LDAP registry
  - Federated LDAP registries
  - SAF registry for z/OS
  - Custom user registry
- Integration with a third-party security service by using trust association interceptors (TAI)
- A custom Java Authentication and Authorization Server (JAAS) login module
- Used for application and JMX security
  - One registry per server

Securing Liberty



*Figure 6-11. User registries*

For authenticating users, Liberty supports the following configurations:

- A basic user registry that defines user and group information for authentication to the Liberty server.
- A Lightweight Directory Access Protocol (LDAP) server that is used for authentication.
- System Authorization Facility (SAF) registry for authorization on z/OS.
- Federated LDAP registries, where two or more LDAP registries are defined so that the operations, such as a search for a user, are executed on all the registries.
- Custom user registries that are installed as an extension to Liberty.
- Integration with a third-party security service by using trust association interceptors (TAIs). A TAI is used to validate HTTP requests between a third-party security server and a Liberty server. The TAI can be called before or after single sign-on (SSO).
- SSO so that web users can authenticate once when accessing Liberty resources such as HTML, JSP files, and servlets. Users can also authenticate once when accessing resources in multiple Liberty servers that share Lightweight Third Party Authentication (LTPA) keys.
- A custom Java Authentication and Authorization Service (JAAS) login module to make more authentication decisions or to make finer-grained authorization decisions inside an application.

## Setting up the BasicRegistry

- Add the `appSecurity-2.0` Liberty feature to the `server.xml` file
- User and group names must be unique

```
<basicRegistry id="basic" realm= "WebRealm"
     <user name="Ted" password="{xor}MzorMjo2MQ==" />
          <user name="Jack" password="{xor}MzorMjo2MQ==" />
          <user name="Bill" password="{xor}MC86MSov=" />
</basicRegistry>
```

```
<basicRegistry id="basic" realm="WebRealm"
     <user name="Ted" password="{xor}MzorMjo2MQ==" />
     <user name="jack" password ="{xor}MzorMjo2MQ==" />
     <group name="administratorGroup">
          <member name="Jack" />
          <member name="Bill" />
     </group>
</basicRegistry>
```




*Figure 6-12. Setting up the BasicRegistry*

You can configure a basic user registry in Liberty for authentication. You can use a basic user registry by defining the users and groups information for authentication on the Liberty server.

You must use unique names for your users and groups. You should remove all trailing and leading spaces from the user and group names. If user ID or password contains characters other than US-ASCII, make sure that the file is saved by using UTF-8 character encoding.

If you use WebSphere Application Server Developer Tools for Eclipse, the password is encoded for you automatically.

If you edit the `server.xml` file directly, you can use the `securityUtility encode` command to encode the password for each user.

## Setting up the LDAP user registry

- Authenticate by using an LDAP server
- Supports: IBM Directory Server, Microsoft Active Directory, IBM Lotus Domino, Novell eDirectory, Sun Java System Directory Server, Netscape Directory Server, IBM SecureWay Directory Server

```
<server>
     <featureManager>
          <feature>appSecurity-2.0</feature>
          <feature>ldapRegistry-3.0</feature>
     </featureManager>

     <ldapRegistry host="myldapserver.ibm.com"
     port="389" baseDN="o=ibm,c=us"
     ldapType="IBM Directory Server" />


</server>
```




*Figure 6-13. Setting up the LDAP user registry*

You can configure one or more Lightweight Directory Access Protocol (LDAP) servers in Liberty for authentication. Ensure that your LDAP server is up and running, and that the host name and port number of the LDAP server are already in your known list.

You can use an existing LDAP server for application authentication in Liberty. To do this task, you add the `appSecurity-2.0` feature to the `server.xml` file and specify, in the `server.xml` file, the `ldapRegistry-3.0` feature, and the configuration information for connecting to the LDAP server.

# Setting up the federated LDAP registries (1 of 2)

```
<featureManager>
          <feature>appSecurity-2.0</feature>
          <feature>federatedRegistry-1.0</feature>
</featureManager>


<ldapRegistry host="ldapserver1.city1.ibm.com"
     baseDN="o=mycompany,ou=myou,c=us" port="123" ldapType="IBM
Tivoli Directory Server" name="o=mybaseentry">
</ldapRegistry>


<ldapRegistry host="ldapserver2.city2.ibm.com"
     baseDN="cn=users,dc=secfvt2,dc=city2,dc=ibm,dc=com"
     port="456" ldapType="Microsoft Active Directory"
     bindDN="cn=testuser,cn=users,dc=secfvt2,dc=city2,dc
     =ibm,dc=com"   bindPassword="{xor}KzosKyosOi0vKDs=">
</ldapRegistry>
```




*Figure 6-14. Setting up the federated LDAP registries (1 of 2)*

Using the LDAP registry, you can also federate multiple repositories and run the LDAP operations on two or more registries. Liberty user can configure the LDAP registry federation feature either directly in the `server.xml` file or can configure in the LDAP Registry Federation section in the developer tool. After the configuration of the federated repositories, you can obtain a consolidated result of the federated repositories on any operation that you want to perform. For example, if you want to perform a search operation for all user names that starts with test, you can perform a search across the set of LDAP registries and get the consolidated search result, which can then be sent back to the calling program.

# Setting up the federated LDAP registries (2 of 2)

```
<federatedRepository>
<primaryRealm name="RealmName" delimiter="@"
allowOpIfRepoDown="true">
<participatingBaseEntry name="o=mybaseentry"/>
<participatingBaseEntry
name="cn=users,dc=secfvt2,dc=city2,dc=ibm,dc=com"/>
<uniqueUserIdMapping inputProperty="uniqueName"
outputProperty="uniqueName"/>
<userDisplayNameMapping inputProperty="principalName"
outputProperty="principalName"/>
<uniqueGroupIdMapping inputProperty="uniqueName"
outputProperty="uniqueName"/>
<groupSecurityNameMapping inputProperty="cn"
outputProperty="cn"/>
<groupDisplayNameMapping inputProperty="cn" outputProperty="cn"/>
</primaryRealm>
</federatedRepository>
```




*Figure 6-15. Setting up the federated LDAP registries (2 of 2)*

Specifying the `<federatedRepository>` element is not mandatory to federate multiple LDAP registries because they are federated automatically. If the `<federatedRepository>` element is specified to configure the `participatingBaseEntry` and `primaryRealm` elements, then the user registry operations are performed only on the repositories that are defined in the `primaryRealm` element. You can define the input and output property mappings for different user registry APIs under the `primaryRealm` element.

The `name` attribute of the `participatingBaseEntry` element must be the same as the value of `baseDN` attribute that is specified in the `ldapRegistry` element. As an example, the `baseDN` and `name` attributes are configured for the LDAP registry on the host *ldapserver1.mycity1.mycompany.com*. The value of `baseDN` attribute must be the same as that of sub tree in your LDAP server and the value of `name` attribute must be the name of that sub tree in the federated user registry. It is optional to specify the `name` attribute. By default, the `name` attribute uses the same value as the `baseDN` attribute. If the `name` attribute is specified in the `ldapRegistry` element, then the `name` attribute in the `participatingBaseEntry` element must use the same value as the `name` attribute in the `ldapRegistry` element.

## Setting up the SAF user registry

- Authenticate by using an SAF on z/OS

```
<featureManager>
    <feature>zosSecurity-1.0</feature>
</featureManager>

<safRegistry id="saf"/>
```




*Figure 6-16. Setting up the SAF user registry*

The System Authorization Facility (SAF) registry holds information that is needed to perform security-related functions such as authenticating users and retrieving information about users, groups, or groups that are associated with users. You activate and configure the SAF registry through the `server.xml` file.

Activate the SAF registry service by adding the `zosSecurity-1.0` feature to the `server.xml` file.

Configure web application security features to use the SAF registry service by adding the `appSecurity-1.0` feature.

Configure the SAF registry by using a `safRegistry` configuration element.

The `safRegistry` element has the following attributes: ID and realm. The ID uniquely identifies this registry instance. The ID can be anything that you want, but must be unique among other configured registries such as the basic registry and the LDAP registry.

The realm specifies the security realm that is associated with the SAF registry. If you do not specify a realm, the default is the plex name (ECVTSPLX).

## Configuring LTPA in Liberty

- You can configure a Liberty server to use a specific Lightweight Third Party Authentication (LTPA) keys file, user-defined password, and expiration time
  - LTPA is configured by default when security is enabled for a Liberty server for the first time
- LTPA keys are encrypted with a randomly generated key and a default password of `WebAS` is initially used to protect the keys
  - The password is required when you import the LTPA keys into another server
  - Change the password to protect the security of the LTPA keys

```
<ltpa keysFileName="yourLTPAKeysFileName.keys"
keysPassword="keysPassword" expiration="120"
monitorInterval="5s" />
```




*Figure 6-17. Configuring LTPA in Liberty*

When the LTPA keys are exchanged between servers, the password must match across the servers for Single sign-on (SSO) to work.

The default location of the automatically generated LTPA keys file is `${server.output.dir}/resources/security/ltpa.keys`.

The default expiration timeout is 120 minutes. The expiration value refers to how long the LTPA tokens are valid before they expire.

To enable dynamic reloading of the LTPA keys when copying an LTPA keys file from another server, you can specify a file monitor interval before copying the LTPA keys file. The monitor interval value refers to how often the LTPA keys file is monitored for updates.

Set the `monitorInterval` attribute to check the `lpta.keys` file for key changes to be dynamically reloaded. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), or seconds (s). In the example on the slide, the LTPA keys file is checked for changes to be dynamically reloaded every 5 seconds.

Remember to encode the password within the configuration.

## Single sign-on (SSO) and Liberty

- SSO enables users to log in one place and access applications on other servers without getting prompted again
- To make SSO work
  - The LTPA keys must be exchanged across different Liberty servers
  - The user registries must be the same
  - The token must not have expired
- When a user passes authentication on one of the Liberty servers, authentication information that is generated by the server is transported to the web browser in a cookie
  - The cookie is used to propagate the authentication information to other Liberty servers
- The LTPA is configured and ready for immediate use
  - The default cookie name that is used to store the SSO token is called `ltpaToken2`




*Figure 6-18. Single sign-on (SSO) and Liberty*

With SSO configuration support, web users can authenticate once when accessing Liberty resources such as HTML, JavaServer Pages (JSP) files, and servlets, or accessing resources in multiple Liberty servers that share Lightweight Third Party Authentication (LTPA) keys.

To exchange the LTPA keys, you can copy the `ltpa.keys` file from one server to another and restart the server to use the new LTPA keys. The registries that are used by all the servers that participate in the SSO domain must be the same.

When a user is authenticated in one Liberty server, the SSO token that is created for the user during the authentication process is put in the cookie that is sent to the HTTP client, for example a browser. If there is another request from that client to access another set of applications on a different server, but in the same DNS that was configured as part of the SSO configuration in the first server, the cookie is sent along with the request. The receiving server tries to authenticate the user by using the token in the cookie. If both conditions are met, the receiving server validates the token and creates a subject that is based on the user in this server, without prompting the user to log in again. If the token cannot be validated (for example, it cannot decrypt or verify the token because of LTPA key mismatch), the user is prompted to enter the credential information again. If you want to use a different name for the cookie, you can customize the cookie name by using the `ssoCookieName` attribute of the `<webAppSecurity>` element. If you customize the cookie name, make sure that all the servers that participate in SSO use the same cookie name.

## Configuring SSL

- Add the SSL feature and provide the keystore password

```
<featureManager>
    <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" password="{xor}DFoKyp="/>
```

- You can expand the minimal SSL configuration with more details

```
<featureManager>
    <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" location="keystore.p12"
password="{xor}DFoKyp=" type="PKCS12" />
```




*Figure 6-19. Configuring SSL*

You can configure the Liberty server to provide secure communications between a client and the server. Certificate is generated at server startup.

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.

To configure secure communications, you can either specify a minimal SSL configuration or a detailed SSL configuration in the `server.xml` file. The minimal configuration requires the SSL feature and a keystore entry to be specified only.

SSL client authentication occurs during the connection handshake by using SSL certificates. The SSL handshake is a series of messages that are exchanged over the SSL protocol to negotiate for

connection-specific protection. During the handshake, the secure server requests that the client send back a certificate or certificate chain for the authentication. To enable SSL in Liberty, you add the `ssl-1.0` Liberty feature to the configuration root document file, `server.xml`, along with code of the keystore information for authentication.

By default, the path and file name for the configuration root document file is `<wlp_root>/wlp/usr/servers/server_name/server.xml`. *path_to_liberty* is the location that you installed Liberty on your operating system, and *server_name* is the name of your server.

## Self-signed certificates

- The `securityUtility` can be used to generate a self-signed certificate

```
localuser@wlphost: /opt/wlp/bin
localuser@wlphost:/opt/wlp/bin$ ./securityUtility createSSLCertificate --server
server1 --password=passw0rd
Creating keystore /opt/wlp/usr/servers/server1/resources/security/key.jks

Created SSL certificate for server server1. The certificate is created with CN=
ocalhost,OU=server1,O=ibm,C=us as the SubjectDN.

Add the following lines to the server.xml to enable SSL:

    <featureManager>
        <feature>ssl-1.0</feature>
    </featureManager>
    <keyStore id="defaultKeyStore" password="{xor}Lz4sLChvLTs=" />


localuser@wlphost:/opt/wlp/bin$
```




*Figure 6-20. Self-signed certificates*

You can use the `securityUtility` command to create a default SSL certificate for use by the Liberty configuration.

Open a command line, then go to the directory to the `<wlp_root>/bin` directory. Create an SSL certificate by entering the command that is shown on the slide. If you do not specify a server name or a password, the command does not run. You create a default keystore `key.jks` for the specified server. The keystore file is located under the `/resources/security` directory of the specified server. If a default keystore exists, the command does not run successfully.

## SSL within Liberty

- SSL can be used to secure network traffic for a number of links
  - From the client to the web server
  - From the plug-in to the Liberty server
  - Other network links can also be secured (LDAP and others)
- Various tools can be used to create and manage the necessary keys and keystores
  - Keystores contain digital certificates that are needed for SSL to establish secure communication between two points

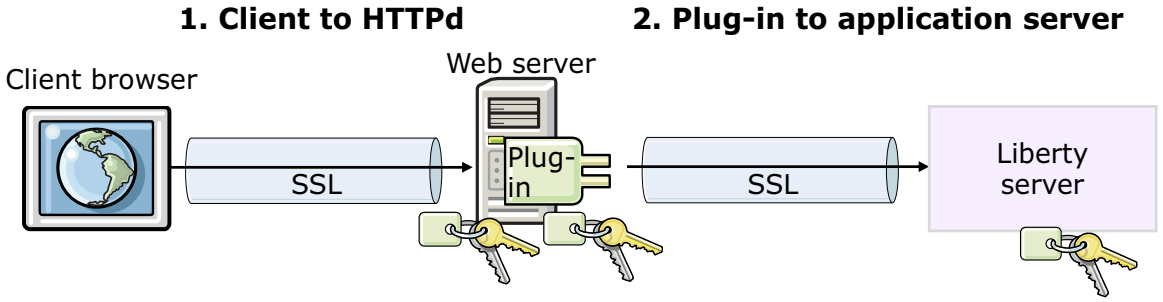






*Figure 6-21. SSL within Liberty*

SSL, or Secure Sockets Layer, provides transport level security between two points, much like a VPN. It is often used to secure communications between a browser and a web server.

SSL can be used to secure communication between clients and web servers, between the WebSphere plug-in and the Liberty server, and between other components in an intranet. You can use the administrative console, or another utility that is called iKeyman, to create and manage the necessary certificates and keystores. A keystore contains the certificates that are needed for SSL to establish sessions between two points.

## Collective security (1 of 2)

- The two principal areas of collective security are
  - Administrative domain security configuration
  - Collective repository data security
- The administrative domain security configuration for collectives is composed of two parts:
  - User domain: This domain relies on Java role-based security that defines the Administrator role, which can be mapped to users within the configured user registry
  - Server domain: This domain relies on SSL certificate-based authentication
- Users in the administrator role can access the MBeans of the collective controller
  - All administrative actions through the collective require that the users are in the administrator role




*Figure 6-22. Collective security (1 of 2)*

You can use the principles of collective security in Liberty to address data in motion and data at rest.

The two principal areas of collective security are:

- Administrative domain security configuration: Addresses data in motion, authentication, and authorization
- Collective repository data security: Addresses data at rest, authentication, and authorization

In order for users to access the collective controller's MBeans, they must be in the Administrator role. All administrative actions through the collective require that the user be granted the Administrator role.

Server-to-server communication falls within the server domain and no user identities or passwords are used to communicate between members of a collective. Each member of the collective has a unique identity within the collective that is composed of its host name, user directory, and server name. Each member within the collective defines its server domain configuration, which consists of the `serverIdentity.jks` and `collectiveTrust.jks` files. These files contain the SSL certificates that are necessary to establish secure communications within the collective. The HTTPS key configuration must have specific trust settings, which are established by default.

The server domain SSL configuration can be customized by adding more trusted certificate entries to the `collectiveTrust.jks` keystore. All trust is copied when a controller is replicated; therefore, SSL customization should be applied to the initial controller. Adding trust to the `collectiveTrust.jks` keystore is only necessary if the default HTTPS certificates are not used.

## Collective security (2 of 2)

- Collective repository data security covers the policy for accessing the contents of the collective repository
  - A collective member is restricted to modifying only its own information in the repository
  - Authenticated administrative users (users in the administrative role) have unrestricted access to information in the repository
- The collective repository ultimately is stored on the disk
  - File system permission settings must be secure for the environment




*Figure 6-23. Collective security (2 of 2)*

The collective repository data security policy covers the policy for data at rest - specifically, the policy of accessing the contents of the collective repository.

The current security policy for collective data is as follows:

The system reserves three node names: `sys.host.auth.info`, `sys.jmx.auth.info`, and `sys.nologin`. These nodes are under a host or server's repository namespace. User-created nodes should avoid using the `sys.` prefix.

The `sys.host.auth.info` and `sys.jmx.auth.info` nodes are not accessible through the MBean to prevent disclosure of system credentials. Accessing the data that is stored at these nodes result in a null response.

A collective member is restricted to modifying only its own information in the repository. Authenticated administrative users have unrestricted access to information in the repository except as previously noted. Authenticated administrative users are all users granted the Administrative role.

Because the collective repository ultimately is on the disk, the file system permission settings must be secure for the environment. It is recommended that the collective controller's configuration be readable and writable only by the user, readable only by the group, and not accessible at all by the

world - in other words, `chmod 0640`. Follow any security guidelines that your organization established.

## Security considerations for Liberty

- Authentication
  - The timeout value for the authentication cache element must be smaller than the expiration value for the LTPA token
- Authorization
  - If you specify an auth-constraint with no roles in an application, then no one is allowed to access the resource
  - Be cautious when you specify the EVERYONE special subject
- Passwords
  - Encode all passwords
- LTPA
  - Verify that all servers use the same LTPA keys
  - Protect file access to the LTPA keys file
  - Make sure that all servers have their date and time synchronized




*Figure 6-24. Security considerations for Liberty*

Protect file access to the LTPA keys file because it contains the cryptographic material that is used to encrypt and decrypt the user data. Ensure that only the server and administrators have access to this file.

Ensure that all servers use the same LTPA keys. In addition, make sure that the all the servers have their time and date synchronized.

When you specify a password, ensure that it is the same password for all servers that use the same set of LTPA keys. The password is not used to generate the keys, but rather it is used to encrypt the LTPA keys file to prevent the keys from being read. If you copy the LTPA keys file to another Liberty server to achieve Single Sign-On (SSO), the password is required to gain access to the keys in the LTPA keys file.

IBM Training

## Unit summary

- Describe basic Liberty security
- Describe authorization in Liberty
- Describe how to configure user registries
- Describe LTPA and SSO in Liberty
- Describe how to enable SSL communication in Liberty




*Figure 6-25. Unit summary*

IBM Training

## Review questions

1. True or False: True or False: A custom Java Authentication and Authorization Server (JAAS) login module can be used for a user registry.

2. True or False: The `securityUtility` can be used to encode passwords and generate a self-signed certificate.

3. True or False: LTPA keys are encrypted with a randomly generated key and a default password of `WebAS` is initially used to protect the keys.




*Figure 6-26. Review questions*

Write your answers here:

1.

2.

3.

## Review answers

1. True or False: True or False: A custom Java Authentication and Authorization Server (JAAS) login module can be used for a user registry.
   The answer is True.

2. True or False: The `securityUtility` can be used to encode passwords and generate a self-signed certificate.
   The answer is True.

3. True or False: LTPA keys are encrypted with a randomly generated key and a default password of `WebAS` is initially used to protect the keys.
   The answer is True.




*Figure 6-27. Review answers*

IBM Training

# Exercise: Using the IBM HTTP Server with SSL to a Liberty server




*Figure 6-28. Exercise: Using the IBM HTTP Server with SSL to a Liberty server*

IBM Training

## Exercise objectives

- 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




*Figure 6-29. Exercise objectives*

# Unit 7. Course summary

## Estimated time

00:15

## Overview

This unit summarizes the course and provides information for future study.

IBM Training

## Unit objectives

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study




*Figure 7-1. Unit objectives*

IBM Training

## Course objectives

- Describe the WebSphere Liberty Profile architecture
- Create a Liberty profile server
- Use the Admin Center to manage Liberty servers
- Deploy clusters of Liberty servers
- Use the collective controller
- Use Jython scripts to administer Liberty servers
- Configure Dynamic Routing
- Configure the auto scaling feature and define auto scaling policies
- Configure SSL communication in Liberty
- Use the IBM HTTP and web server plug-in with Liberty servers




*Figure 7-2. Course objectives*

IBM Training

## References

- The online version of the WebSphere Application Server Liberty Network Deployment IBM Knowledge Center is a good source of product information
  - http://www.ibm.com/support/knowledgecenter/SSAW57_liberty/as_ditamaps/was900_welcome_liberty_ndmp.html
- Numerous articles that are listed in IBM developerWorks
  - https://developer.ibm.com/wasdev/
- IBM WebSphere Application Server V8.5 Administration and Configuration Guide for Liberty Profile:
  - http://www.redbooks.ibm.com/abstracts/sg248170.html




*Figure 7-3. References*

IBM Training

# Enhance your learning with IBM resources

*Keep your IBM Cloud skills up-to-date*

- IBM offers resources for:
  - Product information
  - Training and certification
  - Documentation
  - Support
  - Technical information

- To learn more, see the IBM Cloud Education Resource Guide:
  - www.ibm.biz/CloudEduResources




*Figure 7-4. Enhance your learning with IBM resources*

IBM Training

## Unit summary

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study




*Figure 7-5. Unit summary*

IBM Training

## Course completion

**You have completed this course:**

*Administering WebSphere Application Server Liberty Profile*

**Any questions?**




*Figure 7-6. Course completion*

# Appendix A. List of abbreviations

## A

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **API** | application programming interface |
| **ASCII** | American Standard Code for Information Interchange |

## B

## C

| | |
|---|---|
| **CMS** | Certificate Management System |
| **CPU** | central processing unit |

## D

| | |
|---|---|
| **DB** | database |
| **DHCP** | Dynamic Host Configuration Protocol |
| **DN** | distinguished name |
| **DNS** | Domain Name System |

## E

| | |
|---|---|
| **EAR** | enterprise archive |
| **EE** | Enterprise Edition |
| **EJB** | Enterprise JavaBean |
| **ESB** | Enterprise service bus |

## F

## G

| | |
|---|---|
| **GB** | gigabyte |
| **GUI** | graphical user interface |

## H

| | |
|---|---|
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPD** | HTTP Daemon |
| **HTTPS** | HTTP over SSL |

## I

| | |
|---|---|
| **IaaS** | infrastructure as a service |

| | |
|---|---|
| **IBM** | International Business Machines Corporation |
| **IE** | Internet Explorer |
| **IHS** | IBM HTTP Server |
| **IIM** | IBM Installation Manager |
| **I/O** | input/output |
| **IP** | Internet Protocol |
| **IPSEC** | IP Security |

## J

| | |
|---|---|
| **J2EE** | Java 2 Enterprise Edition |
| **J2SE** | Java 2 Platform Standard Edition |
| **JAAS** | Java Authentication and Authorization Server |
| **JAR** | Java archive |
| **Java EE** | Java Platform, Enterprise Edition |
| **JDK** | Java Development Kit |
| **JKS** | Java Keystore |
| **JMX** | Java Management Extensions |
| **JPA** | Java Persistence API |
| **JPG** | Graphics file type or extension (lossy compressed 24-bit color image storage format developed by the Joint Photographic Experts Group) |
| **JRE** | Java Runtime Environment |
| **JSP** | JavaServer Pages |
| **JSSE** | Java Secure Sockets Extension |
| **JVM** | Java virtual machine |

## K

| | |
|---|---|
| **KB** | knowledge base |

## L

| | |
|---|---|
| **LDAP** | Lightweight Directory Access Protocol |
| **LTPA** | Lightweight Third Party Authentication |
| **LTS** | long-term support |

## M

| | |
|---|---|
| **MB** | megabyte |
| **MBean** | Managed Bean (Managed Java object) |

## N

| | |
|---|---|
| **ND** | network deployment (IBM Liberty WebSphere Application Server Network Deployment) |

## O

| | |
|---|---|
| **OS** | operating system |
| **OSGi** | Open Service Gateway initiative (originally). A component framework for Java |

## P

| | |
|---|---|
| **PaaS** | Platform as a Service |
| **PMR** | problem management record |
| **PMT** | Program Management Tool |

## Q

## R

| | |
|---|---|
| **RAM** | random access memory |
| **RDN** | relative distinguished name |
| **REST** | Representational State Transfer |
| **RXA** | Remote Execution and Access |

## S

| | |
|---|---|
| **SAF** | System Authorization Facility |
| **SAML** | Security Assertion Markup Language |
| **SDK** | software development kit |
| **SOAP** | A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol) |
| **SSH** | Secure Shell |
| **SSL** | Secure Sockets Layer |
| **SSO** | single sign-on |

## T

| | |
|---|---|
| **TAI** | trust association interceptors |
| **TCP** | Transmission Control Protocol |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |
| **TLS** | Transport Layer Security |

## U

| | |
|---|---|
| **UI** | user interface |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Locator |
| **USB** | Universal Serial Bus |
| **UTF** | Unicode Transformation Format |

## V

| | |
|---|---|
| **VM** | virtual machine |
| **VMM** | virtual member manager |
| **VPN** | virtual private network |

## W

| | |
|---|---|
| **WAR** | web archive |
| **WCT** | WebSphere Customization Toolbox |
| **WS** | web services |
| **WSDL** | Web Services Description Language |

## X

| | |
|---|---|
| **XML** | Extensible Markup Language |

## Y

## Z

| | |
|---|---|
| **z/OS** | zSeries operating system |

# Appendix B. Resource guide

Completing this IBM Training course is a great first step in building your IBM Middleware skills. Beyond this course, IBM offers several resources to keep your Middleware skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites.

## Training

- **IBM Training website**
  - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
  - For more information, see: http://www.ibm.com/training
- **IBM Training News**
  - Review or subscribe to updates from IBM and its training partners.
  - For more information, see: https://www.ibm.com/blogs/ibm-training
- **IBM Certification**
  - Demonstrate your mastery of IBM Middleware to your employer or clients through IBM Professional Certification. Middleware certifications are available for developers, administrators, and business analysts.
  - For more information, see: http://www.ibm.com/certify
- **Training paths**
  - Find your next course easily with IBM training paths. Training paths provide a visual flow-chart style representation of training for many IBM products and roles, including developers and administrators.
  - For more information, see: http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=page&c=a0003096

## Social media links

Connect with IBM Middleware Education and IBM Training, and learn about the latest courses, certifications, and special offers by seeing any of the following social media websites.

- **Twitter**
  - Receive concise updates from Middleware Education a few times each week.
  - Follow Middleware Education at: twitter.com/IBMCloudEdu

- **Facebook**:
  - Follow IBM Training on Facebook to keep in sync with the latest news and career trends, and to post questions or comments.
  - Find IBM Training at: facebook.com/ibmtraining
- **YouTube**:
  - See the IBM Training YouTube channel to learn about IBM training programs and courses.
  - Find IBM Training at: youtube.com/IBMTraining

# Support

- **Middleware Support portal**
  - The Middleware Support website provides access to a portfolio of downloadable support tools, including troubleshooting utilities, product updates, drivers, and Authorized Program Analysis Reports (APARS). The Middleware Support website also provides links to online Middleware communities and forums for collaboratively solving issues. You can now customize the IBM Support website by adding or deleting portlets to show the most important information for the IBM products that you work with.
  - For more information, see: http://www.ibm.com/software/websphere/support
- **IBM Support Assistant**
  - The IBM Support Assistant is a local serviceability workbench that makes it easier and faster for you to resolve software product issues. It includes a desktop search component that searches multiple IBM and non-IBM locations concurrently and returns the results in a single window, all within IBM Support Assistant.
  - IBM Support Assistant includes a built-in capability to submit service requests; it automatically collects key problem information and transmits it directly to your IBM support representative.
  - For more information, see: http://www.ibm.com/software/support/isa
- **IBM Education Assistant**
  - IBM Education Assistant is a collection of multimedia modules that are designed to help you gain a basic understanding of IBM software products and use them more effectively. The presentations, demonstrations, and tutorials that are part of the IBM Education Assistant are an ideal refresher for what you learned in your IBM Training course.
  - For more information, see: http://www.ibm.com/software/info/education/assistant/

# Middleware documentation and tips

- **IBM Redbooks**
  - The IBM International Technical Support Organization develops and publishes IBM Redbooks publications. IBM Redbooks are downloadable PDF files that describe

installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines for many Middleware products. Often, Redbooks include sample code and other support materials available as downloads from the site.

  - For more information, see: http://www.ibm.com/redbooks

- **IBM documentation and libraries**
  - IBM Knowledge Centers and product libraries provide an online interface for finding technical information on a particular product, offering, or product solution. The IBM Knowledge Centers and libraries include various types of documentation, including white papers, podcasts, webcasts, release notes, evaluation guides, and other resources to help you plan, install, configure, use, tune, monitor, troubleshoot, and maintain Middleware products. The Knowledge Center and library are located conveniently in the left navigation on product web pages.
- **developerWorks**
  - IBM developerWorks is the web-based professional network and technical resource for millions of developers, IT professionals, and students worldwide. IBM developerWorks provides an extensive, easy-to-search technical library to help you get up to speed on the most critical technologies that affect your profession. Among its many resources, developerWorks includes how-to articles, tutorials, skill kits, trial code, demonstrations, and podcasts. In addition to the Middleware zone, developerWorks also includes content areas for Java, SOA, web services, and XML.
  - For more information, see: http://www.ibm.com/developerworks

# Services

- IBM Software Services for Middleware are a team of highly skilled consultants with broad architectural knowledge, deep technical skills, expertise on suggested practices, and close ties with IBM research and development labs. The Middleware Services team offers skills transfer, implementation, migration, architecture, and design services, plus customized workshops. Through a worldwide network of services specialists, IBM Software Service for Middleware makes it easy for you to design, build, test, and deploy solutions, helping you to become an on-demand business.
- For more information, see: http://www.ibm.com/services/us/en/it-services/systems/middleware-services/

IBM Training