

# Install a Local Java EE Development Environment for Bluemix

---

*Bobby Woolf, IBM Cloud Technical Enablement Specialist*

Learn how to configure a local development environment for developing and deploying Java and Java EE applications for IBM Bluemix. This will customize a typical Eclipse and Liberty development environment.

## Introduction

WebSphere Liberty profile (WLP, a.k.a. WAS Liberty) is the next generation architecture of WebSphere Application Server. The Java Platform, Enterprise Edition (Java EE) runtimes in Bluemix—the Liberty for Java instant runtime and the `ibmliberty` container image—are Liberty runtimes. To develop applications for Bluemix, you need a local development environment that is configured like the Bluemix runtime. This article shows you how—starting with a typical local development environment built using Eclipse and Liberty—to customize that environment for Bluemix and deploy applications to Bluemix. With this guidance, you'll have a functioning development environment and can create Liberty applications to deploy to Bluemix.

This article is the third in a series on installing a local development environment using Eclipse and Liberty. They are:

1. “Quick Introduction to WebSphere Liberty for Java EE Developers” – Background information on how Liberty works and how it differs from other Java EE application servers
2. “Install a Local Java EE Development Environment for WebSphere Liberty” – Instructions for how to install a local environment for developing Java and Java EE applications using Liberty and Eclipse
3. “Install a Local Java EE Development Environment for Bluemix” – This article

This article assumes the reader is familiar with the previous articles.

Readers can use this series to become familiar with Liberty, install a development environment for creating Liberty applications, and deploy those applications to Bluemix. The first two articles apply to all Liberty developers, while the third applies specifically to those using Bluemix.

This article explains the how to configure a Liberty profile for use with Bluemix in two parts:

1. **Configure the local server** – This is the main task for customizing the server configuration in a local Eclipse and Liberty development environment to develop applications to be deployed in the Liberty for Java instant runtime.

2. **Using Liberty and Eclipse with Bluemix** – This documents concepts for how the profile in a Liberty for Java runtime is configured and the Java EE and other features that can be enabled. It also explains common tasks for using Eclipse with Bluemix.

The sections are modular so that you can use just the sections that you find helpful.

## Configure local server for Bluemix

Here is how to configure a local Liberty server to make it suitable for developing applications that can be successfully deployed to the Liberty for Java instant runtime in Bluemix.

First, you'll need a local development environment consisting of the Eclipse IDE and the Liberty profile. If you need to create one, follow the instructions in the article "Install a Local Java EE Development Environment for WebSphere Liberty." By following this process, you will:

- Install a local Liberty profile runtime environment for running Java EE 7 applications
- Install an Eclipse IDE for developing Java EE applications
- Connect Eclipse to the Liberty runtime environment and server so that Eclipse can deploy applications to Liberty

None of this is Bluemix-specific. These steps are a general procedure for setting up Eclipse with Liberty for developing traditional (i.e. non-cloud) Java EE applications.

Second, once you have your Eclipse with Liberty development environment installed, let's configure it for Bluemix. You can configure your local Liberty server for one of three Java EE platforms included with Liberty for Java:

- Java EE 7 Web profile
- Java EE 7 (Full platform)
- Java EE 6 Web profile

You can then individually add any or all of several dozen additional features that are included with Liberty for Java:

- Miscellaneous features

Choose the configuration that is most suitable for your application.

The section "Liberty for Java runtime features" includes Table 1, which lists the Liberty features available in the Liberty for Java runtime. It also shows the convenience features that correspond to the supported Java EE platforms, as well as the Miscellaneous features.

## Liberty version for development

You want your local Liberty development environment to be the same as the Liberty for Java runtime. This will help avoid compatibility issues that might cause your application to function differently in Bluemix than it does locally. So how do you set up your local environment to be like Bluemix?

As of this writing, the Liberty buildpack incorporates Liberty v8.5.5.x and JRE 8, part of Java SE 8; see [Upcoming Liberty for Java buildpack changes](#). As long as your development environment incorporates recent builds of those versions, you'll be in pretty good shape. When Bluemix announces support for Liberty v9, you'll need a runtime for that.

The developers of the Liberty buildpack strive to incorporate the latest versions of Liberty profile and the JRE. If you want to install a runtime that is even more closely aligned with the latest buildpack, install your runtime to match the latest details described in [Latest updates to the Liberty buildpack](#). It specifies the latest version of the buildpack and what version of the Liberty profile and of the IBM JRE 8 the buildpack uses.

## Java EE 7 Web Profile

The Liberty for Java instant runtime is configured, by default, for the Java EE 7 Web Profile. This means that a Java application (a WAR or EAR file) can be deployed (pushed, in Cloud Foundry terminology) and Liberty for Java will run that application in a Java EE 7 Web Profile container. Since that is the default for Liberty for Java, let's configure our development server the same way.

"Enable features" in "Install a Local Java EE Development Environment for WebSphere Liberty" explains how to enable a feature in a Liberty server's configuration. As shown in Table 1, the Liberty convenience feature that corresponds to the Java EE 7 Web Profile specification is named `webProfile-7.0`, which of course is one of the features included in Liberty for Java. To enable that feature, edit your local server's configuration to include this:

```
<featureManager>
  <feature>webProfile-7.0</feature>
</featureManager>
```

With this line, the server's feature manager enables the `webProfile-7.0` convenience feature, which enables all of the features shown in the Java EE 7 Web Profile column in Table 1. Because this is the default configuration of the server in Liberty for Java, an application that runs with these features locally can be deployed to Bluemix without having to change the list of features enabled in Liberty for Java.

## Liberty for Java configuration details

While configuring your local server to enable `webProfile-7.0` is a good approximation of the default configuration of the Liberty server in the Bluemix runtime, the details are a bit more complex. [Options for pushing Liberty applications](#)

shows the explicit list of Liberty features enabled by default and shows the default `server.xml` file. This list is missing a couple of the Java EE 7 Web Profile features and adds a couple that are Bluemix-specific. Despite these differences, your Java EE 7 Web Profile applications will deploy properly.

### Java EE 7 Full Platform

If your application uses Java EE 7 features beyond those included as parts of the Web Profile, you'll need to enable the feature for Java EE 7 (full platform), which is the feature named `javaee-7.0`. You could enable just the features your application needs, but to get started, it's simplest to just enable all of Java EE 7. You could create a new server and configure it this way, but often it's simpler just to update the configuration of your existing server. Either way, to enable the feature, edit your local server's configuration to include this:

```
<featureManager>
  <feature>javaee-7.0</feature>
</featureManager>
```

With this line, the server's feature manager enables the `javaee-7.0` convenience feature, which enables all of the features shown in the Java EE 7 Full Platform column in Table 1. The assets for these features are already installed in the profile because when we downloaded and installed the Liberty profile, we downloaded the build that contains Java EE 7 Full Platform, so the profile installed locally contains the assets for all of the Java EE 7 features. Since all of the features are already installed in the profile, the server just needs to enable the features so that its applications can use them.

However, applications that require this server configuration will not run correctly in Liberty for Java unless you also update that runtime server's configuration. To deploy the application with its server's configuration, you will need to package the application with its server and push the packaged server to Bluemix, or one of the alternative approaches, as described in "Pushing a packaged server to Bluemix."

### Java EE 6 Web Profile

An older application developed for the Java EE 6 Web Profile set of features may run better using those than using the corresponding features in Java EE 7 Web Profile. In that case, configure a new or existing server to enable the `webProfile-6.0` feature:

```
<featureManager>
  <feature>webProfile-6.0</feature>
</featureManager>
```

With this line, the server's feature manager enables the `webProfile-6.0` convenience feature, which enables all of the features shown in the Java EE 6 Web Profile column in Table 1. The Liberty profile we installed contains the Java EE 7 Full Platform assets, but those do not include all of the Java EE 6 Web Profile assets. To

make sure your profile has all of the Java EE 6 Web Profile assets installed, download the `webProfile-6.0` asset (and its component assets) as described in “Download asset” in “Install a Local Java EE Development Environment for WebSphere Liberty.” The profile will download and install any of those assets that it doesn’t already contain.

Since `webProfile-6.0` is not the default server configuration in Liberty for Java, to deploy this application to Bluemix, you will need to package the application archive with its server configuration and push the packaged server to Bluemix, or one of the alternative approaches, as described in “Pushing a packaged server to Bluemix.”

### Miscellaneous features

To use any of the features in the Miscellaneous column in Table 1, you’ll need to follow two steps:

1. Download the asset and install it in the profile. See the section “Download asset” in “Quick Introduction to WebSphere Liberty for Java EE Developers.”
2. Enable the feature in the server configuration. See the section “Enable features” in “Quick Introduction to WebSphere Liberty for Java EE Developers.”

Since these features are not enabled by default in Liberty for Java, to deploy this application to Bluemix, you’ll need to push the packaged server, or use one of the alternative approaches. See the section “Pushing a packaged server to Bluemix.”

## Using Liberty and Eclipse with Bluemix

This section explains background information that will help you understand how the Liberty for Java instant runtime works and how that relates to how a Liberty profile works. It also explains how one connects to Bluemix to deploy applications, and how to use Eclipse tooling for Bluemix.

Liberty for Java is implemented in Bluemix using the Cloud Foundry platform. To understand how Liberty for Java works and how to use it to deploy Java applications into Bluemix, it’s helpful to understand the basics of Cloud Foundry. We’ll introduce that here:

- **Cloud Foundry** – What’s hosting the Liberty for Java runtimes?

When deciding how to configure your Liberty server to work like the Liberty for Java runtime, it’s helpful to understand in a bit of detail what features the runtime makes available and how they relate to the Java EE platforms included in the runtime. We’ll explore that here:

- **Runtime features** – The Liberty features that the Liberty for Java runtime makes available for servers to enable

Before working with Bluemix, it’s helpful to understand how applications are deployed to Bluemix:

- **Bluemix deployment** – How to connect to Bluemix, and how to specify the deployment location and application deployment options

With these concepts in mind, IBM provides some tooling for Eclipse for performing common tasks with Bluemix:

- **Bluemix tools** – How to install the tooling for Eclipse that makes it easier to use with Bluemix
- **Link Eclipse to Bluemix** – How to connect Eclipse to your orgs and spaces in Bluemix
- **Pushing a server** – How to use Eclipse to deploy an application, perhaps including its server configuration, into Bluemix

You only need to read the sections you think will be helpful. Many are for reference, so read them later when you need them.

### Cloud Foundry runtimes

Deployers—the staff that deploys applications to the cloud, including developers who want to test their code in the cloud—need to understand how a cloud works in order to specify how the applications should be deployed and hosted. The cloud isn't just a big bucket you dump applications into; it organizes and configures the applications so that it can manage them. A cloud's underlying structure drives how the deployer will work with it.

The instant runtimes in Bluemix are hosted on the Cloud Foundry platform. *Cloud Foundry* is an open source cloud Platform as a Service (PaaS) on which developers can build, deploy, run, and scale applications. With this infrastructure, you develop and manage only your application code, and Bluemix takes care of the management and maintenance of the infrastructure that powers those apps. (For more details, see [Bluemix Instant Runtimes, Containers or Virtual Machines?](#)) Each Bluemix region hosts a separate Cloud Foundry cloud. Cloud Foundry supports application runtimes for multiple languages and manages running them in a multitenant cloud environment so that new runtimes can easily be created and deleted as needed.

The Liberty for Java instant runtime is implemented by the *Liberty buildpack*. A *buildpack* is a Cloud Foundry artifact, a set of code that creates a language-specific runtime, such as a Java application server. Bluemix includes buildpacks for Java, Node.js, Python, and other languages. Each time the Liberty buildpack is run, it creates a new Java EE runtime that contains its own Liberty profile and server.

Deployment requires deployment tools and creates the opportunity for deployment options. The main set of deployment tools for Cloud Foundry is the *Cloud Foundry command line interface* (CF CLI). Like any CLI, it runs on a client (such as your laptop), connects remotely to the server (the Cloud Foundry cloud), and uses the server's API to enable you to manage the server. One of the main tasks the CLI is used for is deploying applications. There are options for configuring how each application is deployed. Because the server is a Cloud Foundry cloud, the application deployment options are CF-specific.

## Liberty for Java runtime features

The Liberty for Java instant runtime in Bluemix does not and cannot include the assets for all features in the Liberty profile. Some features simply do not make sense in a cloud server environment. However, as of v2.0 of the buildpack, the runtime does include features for these Java EE platforms:

- Java EE 7 (Full platform)
- Java EE 7 Web profile
- Java EE 6 Web profile
- Several dozen additional features

Table 1 shows the features that are available in the Liberty for Java runtime and which convenience features, if any, include each feature. Use this table to decide which convenience feature to enable in your local server's configuration. It also shows additional, miscellaneous features you can optionally enable. Do not enable any features other than the ones listed here; although they may work in your local server, they won't be available in Bluemix.

The point of this very long table is not for you to memorize the whole thing, but to use it as a reference to determine these details:

- A list of all features available in the runtime
- Determine which Java EE platforms each feature is part of (if any)
- Determine which features are included in a particular Java EE platform

The first column lists all features that are available in Bluemix (see [Liberty features supported in Bluemix](#)). Columns 2-4 represent Java EE platforms; for each, it shows the corresponding Liberty convenience features and which features that includes (as described in [Liberty features](#)). The convenience features correspond to Java EE specifications. The individual Java EE features and their corresponding JSRs are listed in [Java EE 7 programming model support](#).

The last column, miscellaneous, includes features that do not correspond to parts of the supported Java EE platforms. Those are features in these categories:

- Extended Programming Models
- Enterprise OSGi
- Operations
- Java EE 6 Technologies
- Bluemix-specific: `appState-1.0`, `cloudAutowiring-1.0`, and `logAnalysis-1.0`

The convenience features are shown in **bold**. The table is current for Liberty buildpack v2.1.



**Table 1: Features available in the Liberty for Java runtime**

<i>Liberty feature in Liberty for Java runtime</i>	<i>Java EE Platform</i>			<i>Misc.</i>
	<i>Java EE 7 Full Platform</i>	<i>Java EE 7 Web Profile</i>	<i>Java EE 6 Web Profile</i>	
<b>javaee-7.0</b>	✓			
<b>webProfile-7.0</b>	✓	✓		
<b>webProfile-6.0</b>			✓	
appSecurity-1.0				✓
appSecurity-2.0	✓	✓		
appState-1.0				✓
batch-1.0	✓			
batchManagement-1.0				✓
beanValidation-1.0			✓	
beanValidation-1.1	✓	✓		
bells-1.0				✓
blueprint-1.0				✓
cdi-1.0			✓	
cdi-1.2	✓	✓		
cloudAutowiring-1.0				✓
concurrent-1.0	✓			
couchdb-1.0				✓
distributedMap-1.0				✓
<b>ejb-3.2</b>	✓			
ejbHome-3.2	✓			
ejbLite-3.1			✓	
ejbLite-3.2	✓	✓		
ejbPersistentTimer-3.2	✓			
ejbRemote-3.2	✓			



<i>Liberty feature in Liberty for Java runtime</i>	<i>Java EE Platform</i>			<i>Misc.</i>
	<i>Java EE 7 Full Platform</i>	<i>Java EE 7 Web Profile</i>	<i>Java EE 6 Web Profile</i>	
el-3.0	✓	✓		
eventLogging-1.0				✓
j2eeManagement-1.1	✓			
jacc-1.5	✓			
jaspic-1.1	✓			
javaMail-1.5	✓			
jaxb-2.2	✓			
jaxrs-1.1				✓
jaxrs-2.0	✓	✓		
jaxrsClient-2.0	✓	✓		
jaxws-2.2	✓			
jca-1.6				✓
jca-1.7	✓			
jcaInboundSecurity-1.0	✓			
jdbc-4.0			✓	
jdbc-4.1	✓	✓		
jms-1.1				✓
jms-2.0	✓			
jmsMdb-3.1				✓
jmsMdb-3.2	✓			
jndi-1.0	✓	✓	✓	
jpa-2.0			✓	
jpa-2.1	✓	✓		
jsf-2.0			✓	
jsf-2.2	✓	✓		

<i>Liberty feature in Liberty for Java runtime</i>	<i>Java EE Platform</i>			<i>Misc.</i>
	<i>Java EE 7 Full Platform</i>	<i>Java EE 7 Web Profile</i>	<i>Java EE 6 Web Profile</i>	
json-1.0				✓
jsonp-1.0	✓	✓		
jsp-2.2			✓	
jsp-2.3	✓	✓		
ldapRegistry-3.0				✓
localConnector-1.0				✓
logAnalysis-1.0				✓
managedBeans-1.0	✓	✓		
mdb-3.1				✓
mdb-3.2	✓			
mongodb-2.0				✓
monitor-1.0				✓
oauth-2.0				✓
openid-2.0				✓
openidConnectClient-1.0				✓
openidConnectServer-1.0				✓
osgiAppIntegration-1.0				✓
osgiConsole-1.0				✓
osgi.jpa-1.0				✓
restConnector-1.0				✓
requestTiming-1.0				✓
rtcomm-1.0				✓
rtcommGateway-1.0				✓

<i>Liberty feature in Liberty for Java runtime</i>	<i>Java EE Platform</i>			<i>Misc.</i>
	<i>Java EE 7 Full Platform</i>	<i>Java EE 7 Web Profile</i>	<i>Java EE 6 Web Profile</i>	
samlWeb-2.0				✓
servlet-3.0			✓	
servlet-3.1	✓	✓		
sessionDatabase-1.0				✓
sipServlet-1.1				✓
ssl-1.0	✓	✓		
timedOperations-1.0				✓
wab-1.0				✓
wasJmsClient-1.1				✓
wasJmsClient-2.0	✓			
wasJmsSecurity-1.0	✓			
wasJmsServer-1.0	✓			
webCache-1.0				✓
websocket-1.0	✓	✓		
websocket-1.1	✓	✓		
wmqJmsClient-1.1				✓
wmqJmsClient-2.0				✓
wsSecurity-1.1				✓

The Liberty for Java runtime also makes some Liberty beta features available. Those features are **not** listed in Table 1, but consider these additional miscellaneous features. For the list of beta features, see [Liberty beta features available in Bluemix](#).

Keep in mind that a server cannot load incompatible features, so it should be configured to enable only features that are compatible. For more details, see [Supported Java EE 6 and 7 feature combinations](#).

Unlike other Liberty environments (Liberty profile installed on-premise or a Docker container with Liberty installed), additional features cannot be downloaded and

installed in the Liberty for Java runtime. The runtime does not have a command line for you to run `installUtility`. So the set of features installed by the buildpack, listed above, are the only ones available for the server to load.

### Bluemix deployment concepts

To deploy and otherwise manage an application in Bluemix—whether using the command line or the Eclipse tooling, whether using the Liberty for Java runtime or a runtime for another language—you need to understand some details about how applications are deployed to Bluemix. Specifically:

- You must be authorized to use Bluemix
- You must specify a location within Bluemix that will host the application
- You can specify deployment options for how the application is deployed

These details are explained further in the following sections.

### Bluemix account

An *account* authorizes you to use Bluemix, and is how IBM bills you for using Bluemix. An account is owned by a *user*, someone who is authorized to perform operations within Bluemix. To authenticate yourself as an account owner—not too surprisingly—you log in by specifying a username and password, where your username is usually your email address. For more details, see [Managing your account: Users and roles](#).

To use Bluemix, you'll need a Bluemix account. If you do not already have one, you can [register for a free trial account](#).

### Bluemix deployment locations

Although Bluemix is logically one big cloud, it is physically subdivided into multiple clouds and logically partitioned into administrative domains. When you deploy an application, you push it to one of these specific locations.

The deployment location is ultimately what's called a space. But a space is hosted in a region and administered by an organization. Thus a location is specified in three aspects:

**Region** – Each *region* is a separate installation of Bluemix. While a cloud can seem like it has no location, that it's worldwide, a region has a location in a specific data center located in a particular geography. For Bluemix Public, regions include US South and Europe United Kingdom. Each install of Bluemix Dedicated and Bluemix Local is also its own region. Each region has a cf API endpoint that client tools use to connect to the region.

**Organization** – An *organization* is a quota of resources available for deploying and running applications. An organization spans regions so that portions of the quota can be used in separate regions. Each corresponds to a Bluemix account, whose owner is the administrator of the organization. The administrator can add other users to the organization. For a single-user account, the organization is typically named the same as the username.

**Space** – A *space* is a group of runtime artifacts—like applications and services—hosted within a region and belonging to an organization. Separate spaces might represent development lifecycle stages like Test and Production, or different teams who are working on unrelated applications but want to share resources.

Figure 1 shows how spaces might be located within two organizations and two regions.

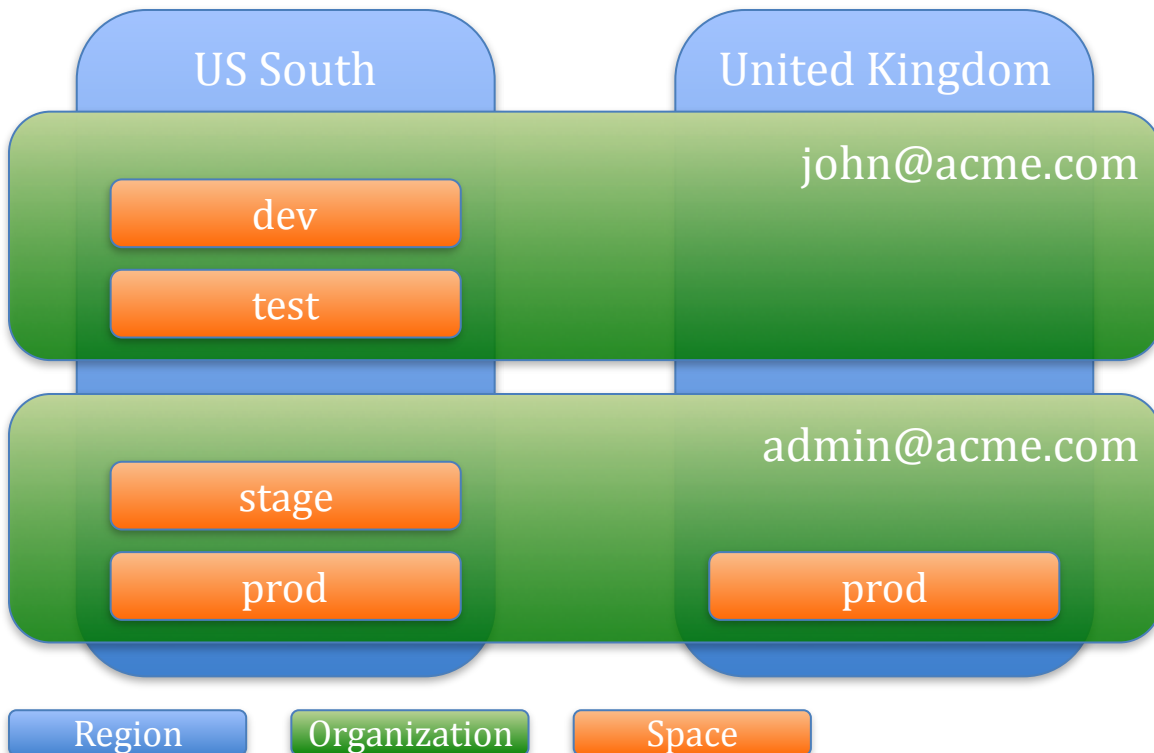


Figure 1: Regions, organizations, and spaces

Figure 1 illustrates how a fictional ACME Corp. might use quotas in Bluemix. The corporation has a developer in the United States named John who has an organization with two spaces, `dev` and `test`, which he uses for developing his parts of applications. The corporation also has an administration team that is responsible for deploying apps into production. That team has spaces for staging as well as production, and production is hosted in two `prod` spaces they'll use to deploy applications active/active across two data centers.

For an overview of Bluemix Public, Dedicated, and Local, see [Bluemix overview: Bluemix architecture](#). For more information about regions, including the cf API endpoint URLs for connecting to the Public regions, see [Bluemix concepts: Regions](#). For more information about organizations and spaces, see [Managing your account: Organizations and spaces](#).

#### Application deployment options

When pushing an application to Cloud Foundry, the push process can be customized with a number of optional settings that specify the application's deployment options: name, memory, host, etc. These can all be specified on the command line or

in a deployment GUI, but often it's convenient to package them with the application in an application manifest file, `manifest.yml`. For details, see [Managing Apps: Deploying apps](#).

That's a quick overview of the Bluemix deployment concepts. We will use these concepts when explaining how to connect the Cloud Foundry CLI or Eclipse to Bluemix and how to deploy applications to Bluemix.

### Install the Bluemix tools

The Bluemix Tools make it easier to use Eclipse to deploy applications to Bluemix, especially the Liberty for Java runtime.

Download and install the Bluemix Tools. The download is available here:

- [IBM Eclipse Tools for Bluemix](#)

You can also download it from the Eclipse Marketplace or WebSphere Software Installer:

- **IBM Eclipse Tools for Bluemix**

The Bluemix Tools are documented here:

- [“Managing applications: Deploying apps with IBM Eclipse Tools for Bluemix”](#) in the Bluemix documentation

The Bluemix Tools are now installed in Eclipse.

### Link Eclipse to Bluemix

You can define servers in Eclipse that link to Bluemix. This enables you to use Eclipse to deploy apps and packaged servers to Bluemix and to interact with the apps that are deployed to Bluemix.

To use Eclipse with Bluemix, you'll need a Bluemix account. If you do not already have one, you can [register for a free trial account](#).

A Bluemix server in Eclipse doesn't represent all of Bluemix, it represents a particular location, specified by a region, organization, and space. Therefore you need one server in Eclipse for each space in Bluemix you want to deploy to. And for each space, you have to authenticate as a user authorized to deploy to that space.

To create a Bluemix server in Eclipse, assuming you want to deploy to your space named “dev”, use the Bluemix tools in Eclipse to perform these steps:

1. Go to the Servers view. (From Eclipse's menu bar, select **Window > Show View > Servers**. See “Open view (such as Runtime Explorer)” in “Install a Local Java EE Development Environment for WebSphere Liberty.”)
2. In the Servers view, from the context menu, select **New > Server**.
3. In the New Server dialog, on the Define a New Server page, as shown in Figure 2:
  - Server type: **IBM > IBM Bluemix**

- Server name: **Bluemix – dev**
- Press **Next**.

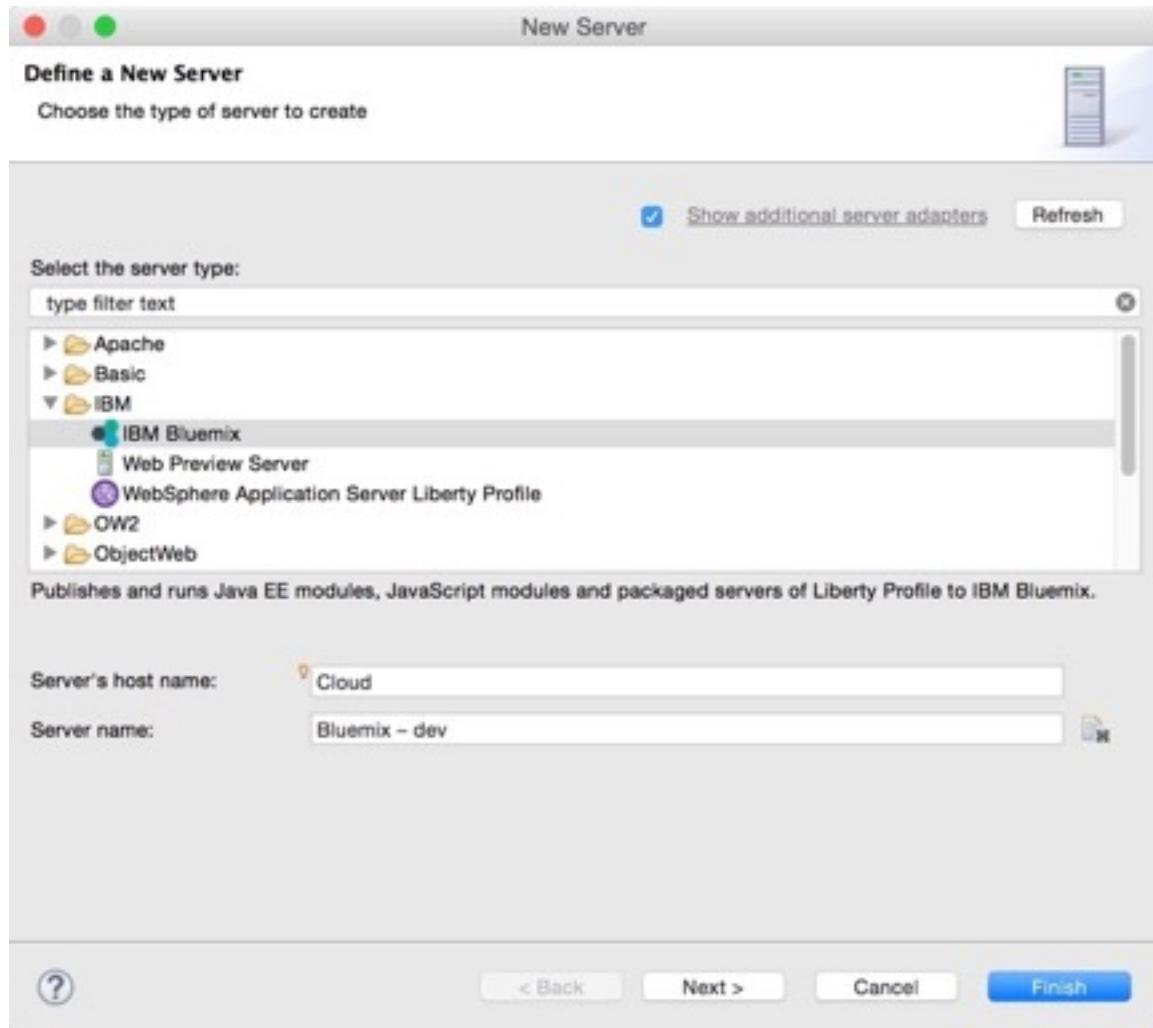
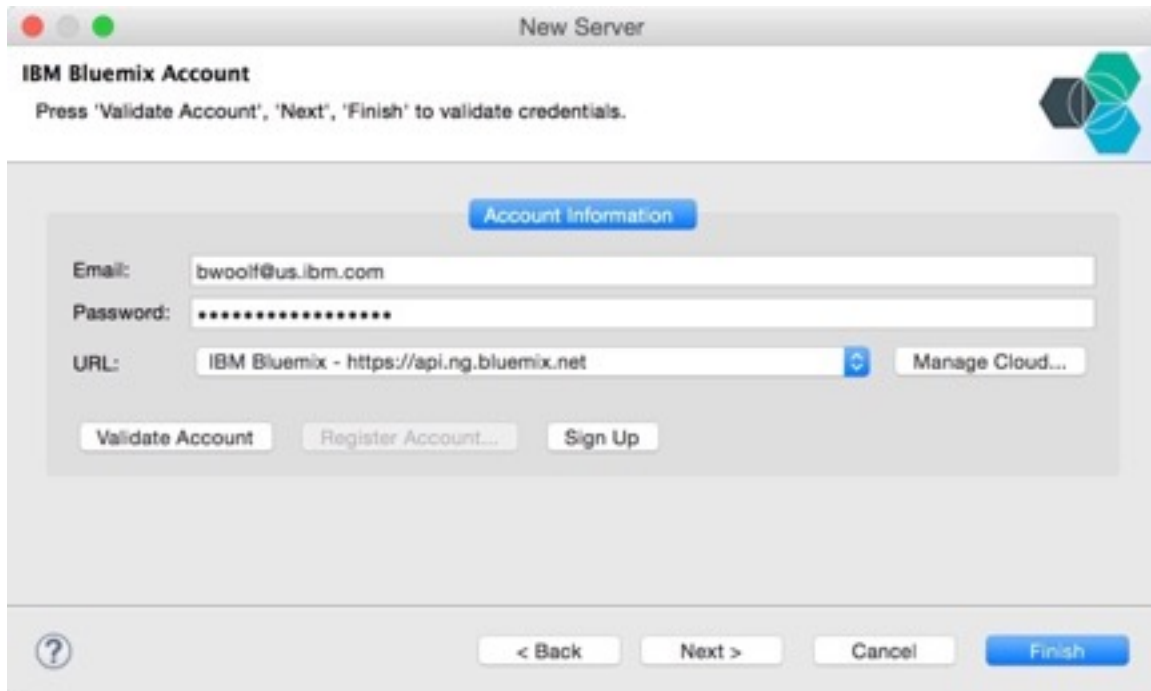


Figure 2: Create new server: Define a new server

4. In the New Server dialog, on the IBM Bluemix Account page, as shown in Figure 3:
  - Provide your credentials to log into Bluemix (authenticating an account that is authorized to deploy to your dev space).
  - Optional: Press **Validate Account** to confirm that your credentials work properly.
  - Also select the region that hosts the space by selecting the URL for the region's cf API endpoint. The default is US South's endpoint.
  - Press **Next**.



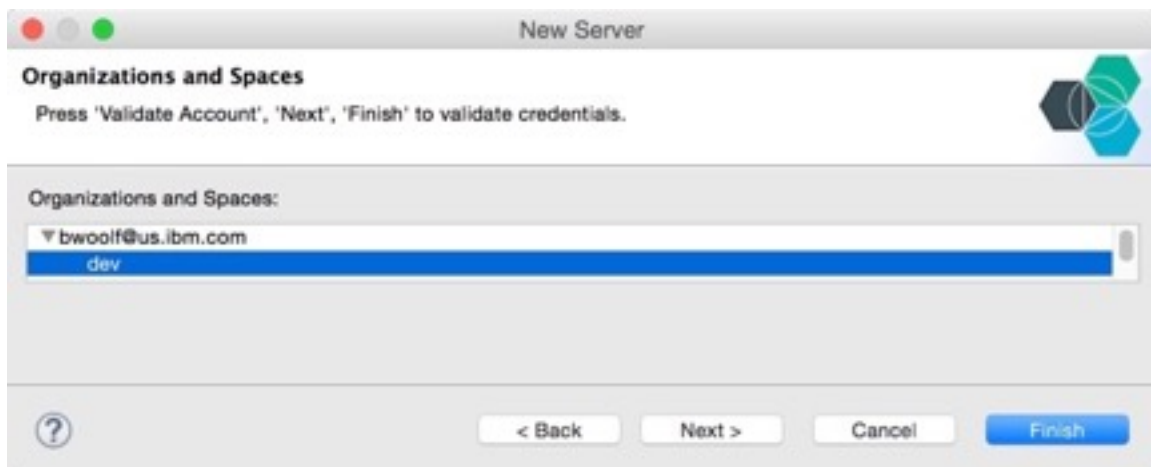


The screenshot shows a macOS-style window titled "New Server". The main heading is "IBM Bluemix Account". Below it, a subtitle reads "Press 'Validate Account', 'Next', 'Finish' to validate credentials." In the top right corner is the IBM Bluemix logo. A blue button labeled "Account Information" is centered above a form. The form contains three fields: "Email:" with the value "bwoolf@us.ibm.com", "Password:" with masked characters, and "URL:" with the value "IBM Bluemix - https://api.ng.bluemix.net". To the right of the URL field is a "Manage Cloud..." button. Below the form are three buttons: "Validate Account", "Register Account...", and "Sign Up". At the bottom of the window are navigation buttons: a help icon (?), "< Back", "Next >", "Cancel", and a blue "Finish" button.

Figure 3: Create new server: IBM Bluemix Account

The wizard uses the region and account to log into Bluemix and retrieve the user's list of organizations and spaces.

5. In the New Server dialog, on the Organizations and Spaces page, as shown in Figure 4:
  - Organizations and spaces: Select your organization and **dev**.
  - Press **Finish**.



The screenshot shows the same "New Server" window, but the main heading is now "Organizations and Spaces". The subtitle remains "Press 'Validate Account', 'Next', 'Finish' to validate credentials." The IBM Bluemix logo is still in the top right. Below the heading is a section titled "Organizations and Spaces:" followed by a dropdown menu. The dropdown shows "bwoolf@us.ibm.com" and a selected item "dev". At the bottom of the window are the same navigation buttons as in Figure 3: a help icon (?), "< Back", "Next >", "Cancel", and a blue "Finish" button.

Figure 4: Create new server: Organizations and spaces

As a result, the Servers view now lists your Bluemix space, as shown in Figure 5.



Figure 5: Servers view listing a Bluemix server

Repeat this process to create a server in Eclipse for each space in your Bluemix org (as well as spaces in other orgs) that you want to deploy to. You can potentially log into each space as a different user.

You can now use the server in Eclipse to perform these tasks in Bluemix:

- [Packaged server support](#) – Push a Liberty server or a packaged server to Bluemix.
- [Pushing an EAR file](#) – Push an application to Bluemix.
- [Incremental Publish](#) – Update application files incrementally without having to repush your entire application.
- [Remote Debug](#) – Run remote debugging against your Java application that is running on Liberty.

Packaged server support is covered in the next section, “Pushing a packaged server to Bluemix.”

### Pushing a packaged server to Bluemix

When you push an application to Bluemix, you only upload the archive file for an application. For a Java application, this is a WAR or EAR file. When you push a server to Bluemix, you not only upload the Java application’s archive file, you also upload the server configuration. This way, settings you configured in your local Liberty server will also be set in the Liberty for Java server in Bluemix.

Before packing a server, to make it as compact as possible, first remove any unneeded features from the feature manager’s list of enabled features.

A server can be pushed using the command line or using the Eclipse Tools for Bluemix. Either way, part of pushing an application is that you need to specify where to deploy the application and the options for deploying the application.

#### Command line

To push a packaged server using the command line, first package the server into an archive such as `defaultServer.zip` (as described in “Packaging a server” in “Install a Local Java EE Development Environment for WebSphere Liberty”), then use the Cloud Foundry command line interface (CF CLI) to push the archive to Bluemix, like this:

```
cf push <yourappname> -p defaultServer.zip
```

You'll need to first log into the Cloud Foundry portion of Bluemix. One of the parameters you need to specify is the cf API endpoint for your region, such as `api.ng.bluemix.net` for Bluemix's US-South region. The form of the login command is:

```
cf login -a <endpoint> -u <user> -o <org> -s <space>
```

To log into Bluemix, you'll need a Bluemix account. If you do not already have one, you can [register for a free trial account](#).

If you don't have the CF CLI, you'll need to download and install it as documented in the resources below.

For Bluemix-specific help with using the CF CLI, see:

- [Deploying your app with the Cloud Foundry command line interface](#)
- [CLI and Dev Tools](#) – How to download
- [cf commands for managing applications](#) – CLI commands

For general Cloud Foundry help with using the CF CLI, see:

- [Developer Guide: cf Command Line Interface \(CLI\)](#) – Installing and using the CLI
- [GitHub: cloudfoundry/cli](#) – Download the CF CLI

By installing the CF CLI and logging into Bluemix, you've now been able to use it to push applications to Bluemix.

### Command line – Alternative approaches

Instead of pushing a packaged server, there are two alternatives for pushing an archive and still customizing the server configuration in the Liberty for Java runtime. These alternative approaches are explained in [Options for pushing Liberty applications](#):

- **Server directory** – If the only changes to the local server's configuration are in its `server.xml` file, or to other files in the `wlp/usr/servers/<server-name>` directory, than rather than packing and pushing the server, you can just push the server directory.
- **Environment variable** – If the only change to the `server.xml` file is to the feature manager's list of enabled features, you can instead specify that list in this environment variable: `JBP_CONFIG_LIBERTY`. When an app is deployed (without a server), the default server configuration will be modified with this list.

You can use these alternative approaches when your changes to the default server configuration in the Liberty for Java runtime are fairly simple.

### Eclipse tools – Bluemix

One of the tasks listed earlier that can be performed with the Bluemix tools is to push a packaged server to Bluemix:

- [Packaged server support](#) – Push a Liberty server or a packaged server to Bluemix.

To push a server using the Eclipse Tools for Bluemix, you don't even need to package the server first. Packaging the server and pushing it to Bluemix is all one task. To perform this task:

1. Go to the Servers view. (From Eclipse's menu bar, select **Window > Show View > Servers**. See "Open view (such as Runtime Explorer).")
2. In the Servers view, select the server you want to package and deploy.
3. Expand the server's tree view to see the Eclipse projects associated with the server. These are the applications that will be packaged with the server, so remove any you don't want to include. (Typically, you include just one application.)
4. With the server selected, from the context menu, select **Utilities > Package Server to IBM Bluemix**.
5. In the Bluemix Server dialog, specify the Bluemix server to deploy the application to, as shown in Figure 6, and press **OK**.



Figure 6: Deploy packaged server

If you do not already have a Bluemix server defined in Eclipse, you need to link Eclipse to one of your spaces in your Bluemix org, as described in "Link Eclipse to Bluemix."

When Eclipse deploys the application to Bluemix, it begins by opening the Application Deployment Options wizard, which enables you to specify the application deployment options for deploying this application. Figure 7 shows the first page of the wizard.

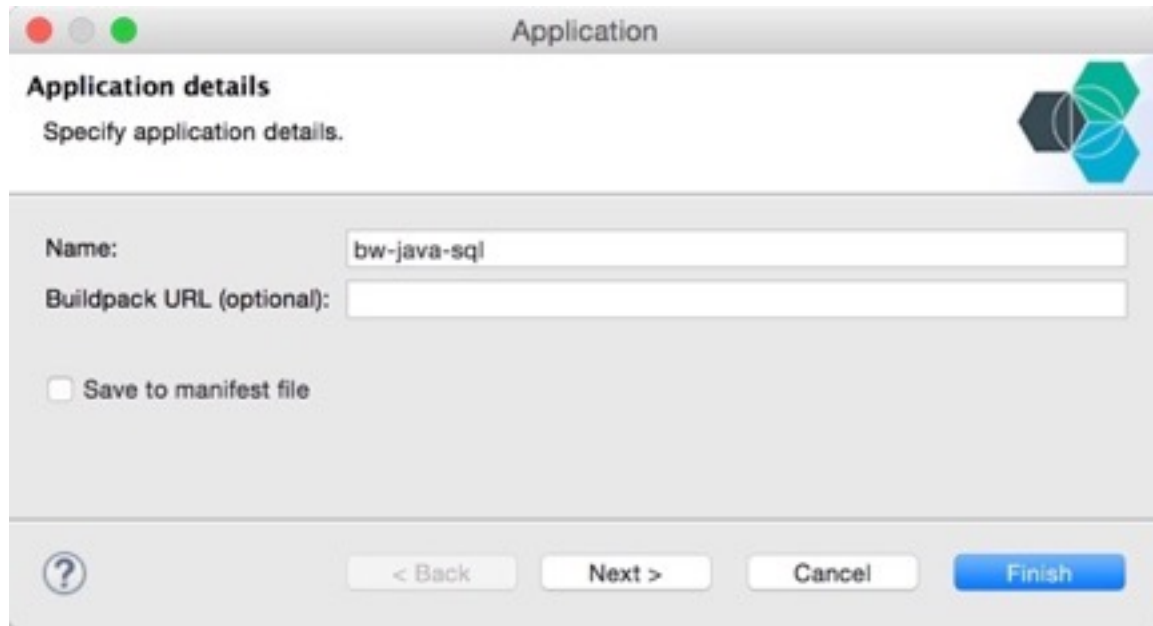


Figure 7: Application deployment options: Application details

If the application has a manifest file, the wizard uses those values as its defaults. Select **Save to manifest file** to save your settings for future deployments.

Once you complete the wizard and press **Finish**, the Console view in Eclipse shows the log as the application is pushed to Bluemix and bound to its services. If any failures occur during the push process, they'll be logged and shown in the console. Once the push completes successfully, you can navigate in the Bluemix dashboard to the target organization and space and you'll see your application listed.

You have now used the Bluemix Tools in Eclipse to deploy an application by pushing it to Bluemix.

### Eclipse tools – Cloud Foundry

If all you want to do is use the CF CLI in Eclipse, there's a tool for that. However, the Bluemix tools do what the CF tools do and more. For example: The Bluemix tools don't just push the server, they package it first.

Nevertheless, to install the CF tools in Eclipse, you can do so from here:

- [Cloud Foundry Integration for Eclipse](#)

Whether you used the command line or Eclipse GUI, you have now pushed a packaged server into Bluemix, which has deployed your application and server configuration to a new Liberty for Java runtime.

## Conclusion

This paper has shown how to configure the Liberty server in the local development environment to work like the server in Bluemix's Liberty for Java runtime. And it has shown you some related concepts and tasks for using Liberty and Eclipse with

Bluemix. With this guidance, you'll be prepared to develop and deploy Liberty applications for Bluemix.

### Acknowledgements

The author would like to thank the following IBMers for their help with this article: David Currie, Jack Cai, Ross Pavitt, Jarek Gawor, Rick Osowski, Pam Geiger, and Dave Thiessen.

### Resources

- [Bluemix Developers Community](#)
  - [Bluemix Registration](#)
  - [Upcoming Liberty for Java buildpack changes](#)
  - [Liberty Buildpack Updates: Java EE 7 is here!](#)
  - [Bluemix Instant Runtimes, Containers or Virtual Machines?](#)
- [IBM Bluemix](#)
- [WebSphere Application Server \(Distributed and IBM i operating systems\), Version 8.5.5 documentation](#)
  - [Liberty profile overview](#)
    - [Java EE 7 programming model support](#)
    - [Liberty features](#)
    - [Supported Java EE 6 and 7 feature combinations](#)
- [Bluemix Docs](#)
  - [Bluemix overview](#)
    - [Bluemix architecture](#)
    - [Bluemix concepts: Regions](#)
  - [Managing applications](#)
    - [Deploying apps](#)
  - [CLI and Dev Tools](#)
    - [Deploying your app with the Cloud Foundry command line interface](#)
    - [cf commands for managing applications](#)
    - [Deploying apps with IBM Eclipse Tools for Bluemix](#)
    - [Managing applications: Deploying apps with IBM Eclipse Tools for Bluemix](#)
    - [Developing with Eclipse Tools](#)

- [Managing your account](#)
  - [Organizations and spaces](#)
  - [Users and roles](#)
- [Creating apps with Liberty for Java](#)
  - [Liberty features supported in Bluemix](#)
  - [Liberty beta features available in Bluemix](#)
  - [Latest updates to the Liberty buildpack](#)
- [WASdev – The WebSphere Application Server Developers Community](#)
  - [IBM Eclipse Tools for Bluemix](#)
  - [IBM WebSphere Liberty Buildpack Contributed to Cloud Foundry](#)
- [Developer Guide: cf Command Line Interface \(CLI\)](#)
- [Cloud Foundry Integration for Eclipse](#)
- [Java Platform, Enterprise Edition \(Java EE\) 7](#)