Constructing applications and application environments is a core part of IT in many enterprises today. Because the process of building such resources are usually human-driven – and can therefore be time-consuming and error-prone -- users naturally turn to technology to help automate these processes. With many automation products available, choosing one can be daunting. This article looks at a simple integration between IBM® WebSphere® CloudBurst™ and IBM Rational® Build Forge® to fully automate such a process, and thereby provide a rapid and repeatable means for delivering applications and application environments.

Introduction

Automating the processes of building and deploying IBM WebSphere Application Server environments and the software that gets deployed to these environments can prove extremely beneficial for organizations of all sizes. By automating the many steps that can potentially be involved, enterprises would benefit from a consistent, repeatable flow that constructs and initializes
application environments with no user intervention. This could accelerate software delivery cycles by enabling development, test, and operations teams to focus on the services they should be providing, rather than the supporting infrastructure that often slows down the delivery of such services.

This article looks at how you can achieve process automation at this level by leveraging a combination of the new IBM WebSphere CloudBurst Appliance with IBM Rational Build Forge and IBM Rational Automation Framework for WebSphere. The simple integration described here takes advantage of key strengths of each product.

**Integration overview**

There are three basic, high level steps involved in setting up a typical WebSphere Application Server environment. Shown in Figure 1, these steps include:

- Installing the operating system.
- Installing the WebSphere Application Server middleware.
- Installing and tuning applications on the middleware.

**Figure 1. Steps to create a WebSphere Application Server environment**

Though each of these steps is required, who performs the step, when the step is performed, and how many times the same step is performed varies with each automation solution. The combination of WebSphere CloudBurst and Rational Build Forge focuses on an integration that minimizes and simplifies the steps you must do.

The first three sub-tasks in the process, as listed in Figure 1:

- Install and configure the operating system
- Install WebSphere Application Server and fixpacks
• Create profiles

are all performed by IBM WebSphere Application Server Hypervisor Edition, which is included with
WebSphere CloudBurst. Therefore, by using WebSphere CloudBurst, these steps are completely
eliminated from your set-up process.

The next step in the process is to install custom software (for example, monitoring agents or
device drivers) into the operating system. This software is typically needed in every application
environment, and changes to this software are infrequent. Therefore, you should add this content
directly to the virtual image deployed by WebSphere CloudBurst so that it only needs to be
installed once: it can be used in a variety of different application environments.

At this point, the process transitions from steps that are common to every virtual machine to steps
that are specific to a given application deployment, such as creating the WebSphere Application
Server topology for a certain application, and then deploying the applications. In other words,
this is the stage where you switch from build-once items for creating a common image, to using
WebSphere CloudBurst patterns and scripts. This enables the same image to be used for different
configurations just by running different scripts during the deployment process.

WebSphere CloudBurst uses metadata-based patterns to describe the construction of a
WebSphere Application Server topology and uses scripts to implement the patterns. These
implementation scripts transparently perform steps, such as federation of WebSphere Application
Server nodes and starting the nodes. This technique enables the same WebSphere Application
Server Hypervisor Edition image to be used for all the different topologies. The pattern
implementation scripts were created by IBM and are included in the appliance. Therefore, by using
WebSphere CloudBurst to automate this process, you do not have to develop scripting for these
common WebSphere Application Server configuration tasks.

Once a topology has been laid down, the application deployment and application-specific
configuration steps are next. As the application owner, you must provide your application-specific
configuration and deployment, and for this scripts are the perfect answer. If these steps were built
directly into the image, you would end up with a lot more images to manage (and images can be
quite large), and the images could change frequently; in fact, a whole new image would be needed
each time an application change is made.

Rather than developing your scripts from scratch and investing in cross script orchestration code,
leveraging Rational Build Forge and the Rational Automation Framework for WebSphere is an
excellent alternative. The steps to build, package, and deploy applications onto a WebSphere
Application Server topology can easily be developed as a Rational Build Forge project. After
WebSphere CloudBurst dispenses a pattern and creates the topology, the Rational Build Forge
project is invoked, as shown in Figure 2. The project packages and deploys the application.
Rational Automation Framework for WebSphere provides a rich set of WebSphere Application
Server configuration scripts to leverage in this process.
To fully minimize the number of images and scripts that are required, WebSphere CloudBurst provides a mechanism for deployment time parameters. Patterns and script packages contain parameters that are specified at deployment time and passed into the image as it is being instantiated. A very common use is to feed in passwords, which typically should be unique to every deployment. WebSphere CloudBurst provides the ability to accept deployment time parameters for script packages also. Later, you will see how this technique is leveraged by the Rational Build Forge integration script.

**Integration details**

Accomplishing this integration is very simple. The WebSphere CloudBurst design includes the ability to specify a user-provided script package to invoke as part of a pattern deployment. The integration is done by creating a customized WebSphere CloudBurst pattern that contains a special script package that interfaces to Rational Build Forge, as shown in step 2 of Figure 3.

Because the integration script is parameterized to take in the Rational Build Forge hostname, project name, and server login credentials, you enjoy a high level of flexibility and reuse. When a pattern is defined that contains the integration script, the values for the parameters can be entered and locked, or left as deployment-time parameters:

- If the parameters are locked, the pattern is directly linked to a specific Rational Build Forge server and project.
- If the parameters are left for deploy time, the same WebSphere Application Server pattern can be reused to integrate with multiple different Rational Build Forge servers and projects.

As with other script packages, the integration script package is executed near the end of the deployment process by default, after all WebSphere Application Servers are started and federation is complete. The integration script first retrieves its input parameters, mentioned above. Running the script package then issues a Java™ Web services call from the virtual machine setup by WebSphere CloudBurst to the specified Rational Build Forge server. This Web services call retrieves the steps from the specified Rational Build Forge project and starts the Rational Build Forge agent in the virtual machine. When the steps have been retrieved, they are run on the virtual machine created by WebSphere CloudBurst. These steps are all shown in Figure 3.
The steps in the Rational Build Forge project are performed on the WebSphere CloudBurst virtual machine, not the Rational Build Forge server. If your steps require any sort of network connectivity, perhaps to retrieve applications or application source code from a remote repository, then you need to make sure your virtual machines have appropriate network access.

**Why integrate these products**

It’s true that you could use either WebSphere CloudBurst or Rational Build Forge by itself to create a customized WebSphere Application Server environment that includes the operating system, WebSphere middleware, and user applications. However, by not integrating these two products together you sacrifice flexibility and functionality.

If you were to use WebSphere CloudBurst in this capacity without Rational Build Forge, script packages would need to be developed and put into the WebSphere CloudBurst pattern to carry out all the application deployment coordination. By enabling Rational Build Forge to handle the application building and deployment process, the actual application becomes more loosely coupled from the WebSphere CloudBurst pattern. This lets you deploy updated versions of applications without requiring updates to the WebSphere CloudBurst patterns, plus gives you the ability to use the same pattern for different applications. Additionally, the Rational Automation Framework for WebSphere provides a set of scripts for common WebSphere Application Server application deployment and configuration that can be used in Rational Build Forge projects, greatly reducing custom scripting requirements. Of course, you could build a super script to mimic all the capabilities of Rational Build Forge and Rational Automation Framework for WebSphere, but why would you want to go through that effort?

On the flip side, given that Rational Build Forge is an adaptive execution environment, you could conceivably include the setup of both the operating system and WebSphere Application Server environments as steps within a project, but running through the installation and configuration process is slower than the image-based techniques leveraged by WebSphere CloudBurst. In addition, WebSphere CloudBurst enables you to control the complete lifecycle of virtualized WebSphere Application Server environments. Using the appliance, you can create, deploy, and manage such environments, and it is very easy to remove the environments once they
are no longer needed, thus reclaiming resources for your cloud. WebSphere CloudBurst also provides WebSphere Application Server intelligence in the form of hardened, best practice configurations built on ten years of field experience. Again, you could write super scripts to mimic all the capabilities of WebSphere CloudBurst, but why would you want to?

Integrating WebSphere CloudBurst and Rational Build Forge leverages each product so that each is used primarily for what it does best, and enables you to benefit from maximum agility, flexibility, and rapidity in both interactions with applications, and with the environments that support those applications.

**Integration scenario**

The remainder of this article describes a step by step integration scenario using WebSphere CloudBurst and Rational Build Forge. To follow the steps described here requires that you have access to a WebSphere CloudBurst Appliance, Rational Build Forge installation, and the sample script package that is included with this article.

1. **Create a Rational Build Forge project**
   To begin this exercise, you will create a project called MyWebProject in Rational Build Forge. This simple project will check out, build, and deploy a sample Web services application to the WebSphere Application Server environment that is provisioned by WebSphere CloudBurst. Your project is made up of four steps:
   a. **Check out source** from a remote code repository.
   b. **Compile source** code of the application.
   c. **Package Web application**, made up of the compiled source, into multiple Java EE WAR modules and a single EAR module. Each module contains the appropriate deployment descriptors.
   d. **Install packaged Web application** to the WebSphere Application Server cell that was deployed by WebSphere CloudBurst.

The first three steps are fairly straightforward, and they make sure that your application source is retrieved and built, and that the application is packaged for deployment to the WebSphere Application Server environment. Step d is worth examining in more detail, as it involves aspects of the WebSphere CloudBurst/Rational Build Forge integration. In step d, notice the source `/etc/virtualimage.properties` statement (Figure 4). The `virtualimage.properties` file is created by WebSphere CloudBurst on each virtual machine and contains information about the WebSphere Application Server environment in the virtual machine and across the cell. The information in this file includes the location of the WebSphere Application Server installation, location of the profile root, cell name, node name, host name, and much more.
Figure 4. Install application from project

You can see in the figure that steps in a Rational Build Forge project can leverage the information in the /etc/virtualimage.properties file. In this case, the step is using the WAS_PROFILE_ROOT variable to invoke the wsadmin tool included as part of the WebSphere Application Server profile in the virtual machine. Here, wsadmin is used to install the application that was built and packaged in steps a, b, and c.

2. **Add a sample script package to WebSphere CloudBurst**

Next, you want to create a custom pattern that includes the script package used for integrating with Rational Build Forge. Again, the script package contains a script that drives communication between WebSphere CloudBurst and Rational Build Forge.

a. Log into the WebSphere CloudBurst console as a user with permission to manage the catalog and create patterns.

b. From the Welcome page, click on **Add script packages**.

c. Click the green cross in the left pane of the Script Packages panel (Figure 5). In the dialog box prompting for a script package name, enter **Rational Build Forge Integration**.

d. Attach the .zip file you downloaded from this article to the new script package. Supply the execution information for the script.
Figure 5. Defining the script package

Defining the script package includes defining four environment variables that will be passed to the script during execution. These four variables provide the script with information it needs to establish the communication between WebSphere CloudBurst and Rational Build Forge:

- **BF_LAUNCH_PROJECT_NAME**: Rational Build Forge project name
- **BF_SERVER_HOST**: Rational Build Forge host server name
- **BF_SERVER_USER**: Username for logging in to Rational Build Forge server
- **BF_SERVER_PWD**: Password for logging in to Rational Build Forge server

You can supply a default value for these variables within the script package definition if you wish. These values can also be supplied at pattern deploy time, or the values can be entered and locked into the pattern so that they have the same value for all deployments of the pattern.

You also need to supply:

- **Working directory** is the directory where the .zip file attached to the script package will be expanded. The script included in the .zip file will therefore be run from this directory.
- **Logging directory** contains any logs produced by invoking the script package.
- **Executable** is the path to the script that should be invoked, which in this case is the script that invokes communication between WebSphere CloudBurst and Rational Build Forge.

After the script package has been defined, you can optionally mark it read-only by clicking the **Lock** icon on the top right toolbar.

Your script package has now been loaded into the WebSphere CloudBurst catalog.

3. **Create a customized WebSphere CloudBurst pattern**

Next, you will create a pattern that will contain the script package:

- a. From the Welcome page in the console, click **Create patterns**.
- b. In this example, you will use a clone of the WebSphere single server pattern as the basis for your custom pattern. To clone the pattern, click on the pattern and select the **Clone**
icon on the top right toolbar. Provide the necessary information for the new pattern in the dialog (Figure 6).

**Figure 6. Creating the pattern**

![Image of creating the pattern](image)

c. To enter edit mode, click on the **Edit** icon on the top right toolbar.

d. In the drop-down menu on the Pattern Editor panel, select **Script Packages**.

e. Drag and drop the Rational Build Forge Integration script package onto the Standalone server component on the canvas. After the script package is added, click the **Done editing** link in the top right corner of the page to return to the Patterns page.

f. At this point, you can optionally lock the pattern in the same manner as the script package.
4. **Deploy the WebSphere CloudBurst pattern**

Your WebSphere CloudBurst pattern with the Rational Build Forge integration script package is now ready to be deployed:

a. From the pattern definition panel, click the **Deploy** button and provide a name for the virtual system.

b. Click on **Configure virtual parts** and then click on **Standalone server**.

**Figure 8. Deploying the pattern**

On the Standalone server configuration panel, WebSphere configuration like cell name, node name, and passwords are supplied (Figure 8). Notice also that values are supplied for the four
variables defined as part of the Rational Build Forge Integration script package. Although these values could have been burned into the pattern, supplying the values at deploy time affords the flexibility to use the same pattern with multiple different Rational Build Forge projects and even different Rational Build Forge servers.

c. When all necessary information has been supplied, click **OK** to begin the deployment process. When the deployment process is complete, the Virtual Systems page will be updated to include the current status of the virtual system.

**Figure 9. Script package execution**

When the script package has completed running, the Rational Build Forge console should show that the project referenced during pattern deployment was successfully executed (Figure 10).

**Figure 10. Rational Build Forge job**

Each of the steps in the MyWebProject project above were run on the virtual machine created for the Standalone server part in your WebSphere CloudBurst pattern.

5. **Verify the results**

To confirm that the project had the intended results -- meaning, in this case, that the sample Web services application was installed -- check the WebSphere Application Server administrative console to make sure that the application appears (Figure 11).
In the figure, notice that the sample Web services application, JaxWSServicesSamples, was installed to the WebSphere Application Server instance created by WebSphere CloudBurst.

More integration

In addition to the deploy-time integration described so far, there are also some interesting extensions:

- **Redeploy an application**
  WebSphere CloudBurst Appliance V1.1 added the capability to specify a user initiated script. Scripts designated as user initiated are not run automatically at the end of deployment. Rather, these scripts are run when a user requests and the invocation can be repeated as many times as desired. By adding the Rational Build Forge integration script to the catalog twice -- once for initial deployment, and again as a user initiated script package -- patterns can be created containing both scripts. This enables you to not only automate the initial application installation, but also to deliver application updates to the environment set up by WebSphere CloudBurst by re-invoking the user-initiated script package. Of course, to ensure that the user-initiated redeploy works correctly, you will want to verify that your Rational Build Forge project includes function to not only install the application, but also uninstall the application if it already exists.

- **Invoke WebSphere CloudBurst from Rational Build Forge**
  The steps in a Rational Build Forge project can do pretty much whatever you want them to do. Therefore, to further automate the process described in this article, you can create a Rational Build Forge project containing steps that utilize either the WebSphere CloudBurst command line interface or REST API to initiate the deployment of the WebSphere CloudBurst pattern that contains the integration script package. Once the deployment is initiated, WebSphere CloudBurst will dispense the pattern, and the integration script will run, invoking a separate Rational Build Forge project. This fully automates the process of deploying a WebSphere CloudBurst pattern and running a Rational Build Forge project that delivers applications into the dispensed environment.

**Summary**

Both IBM WebSphere CloudBurst and IBM Rational Build Forge deliver significant value to users, but when the two products are combined an even higher level of value can be realized. Integrating
these offerings affords you a single, automated process that covers everything from creating a full IBM WebSphere Application Server environment to building and deploying applications to that environment. The ability to create such customized application environments in a repeatable, consistent, and timely way provides significant benefits to all manner of technical teams.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wca-rbf-scriptpackage.zip</td>
<td>850KB</td>
</tr>
</tbody>
</table>
Resources

- IBM WebSphere CloudBurst Appliance product information
- IBM Rational Build Forge product information
- Customizing with WebSphere CloudBurst (series)
  - Part 1: Creating highly customized private clouds
  - Part 2: Using WebSphere CloudBurst to customize a WebSphere middleware environment
  - Part 3: Using script packages for customizing above and beyond patterns
- Cloud computing for the enterprise (series)
  - Part 1: Capturing the cloud
  - Part 2: WebSphere sMash and DB2 Express-C on the Amazon EC2 public cloud
  - Part 3: Using WebSphere CloudBurst to create private clouds
- Space: WebSphere Cloud Computing for Developers
- Space: Cloud Computing Central
- Forum: WebSphere CloudBurst Appliance Forum
- Blog: A view from the clouds: Cloud computing for the WebSphere developer
- Cloud Computing Journal
- Follow us on Twitter
- Is there value in cloud computing?
- IBM developerWorks WebSphere
About the authors

Dustin Amrhein

**Dustin Amrhein** joined IBM as a member of the development team for the WebSphere Application Server. While in that position, Dustin worked primarily on web services infrastructure and web services programming models. In addition, Dustin worked on the development of a RESTful services framework for Java runtimes. In his current role Dustin is a WebSphere Client Technical Professional.

David Brauneis

**David Brauneis** is a senior architect and development lead for IBM Rational Software Delivery Automation Segment including Rational Build Forge and the Chief Architect of the Rational Automation Framework for WebSphere. He joined Rational in February of 2007. Prior to joining the Rational Build Forge team, David spent 8 years on the WebSphere Application Server team in a variety of architecture and development roles. In addition to his normal job responsibility, David was a part of the team that designed and implemented the automated build system used by WebSphere. He has 8 years of WebSphere Application Server development experience, over 7 years of J2EE application development experience, and over 11 years of Java development experience. David has been working for IBM in Research Triangle Park, North Carolina for 10 years on projects related to distributed computing, build, automation, user interface, and system administration. He holds a B.S. in Biomedical Engineering and a Masters in Technical Communication from Rensselaer Polytechnic Institute.

Ruth Willenborg

**Ruth Willenborg** is a Senior Technical Staff Member in IBM's WebSphere Technology Institute. Ruth is currently working on WebSphere Cloud computing and virtual appliance initiatives, and is the technical evangelist for the new IBM WebSphere CloudBurst Appliance. Prior to her work on virtualization and appliance initiatives, Ruth was the manager of the WebSphere Performance team responsible for WebSphere Application Server performance analysis, performance benchmarking and performance tool development. Ruth has over 20 years of experience in software development at IBM. She is co-author of *Performance Analysis for Java Web Sites* (Addison-Wesley, 2002).
Leigh Williamson

Leigh Williamson is an IBM Distinguished Engineer who has been working in the Austin, Texas lab since 1988. Over that time period, Leigh has contributed to many of IBM's major software projects including OS/2, DB2, AIX, OpenDoc, Java, Component Broker, and WebSphere Application Server. Leigh is currently Senior Architect for the Rational Software Delivery Automation line of products including Build Forge, Software Analyzer, AppScan and Rational Automation Framework for WebSphere.

As a member of the Core Rational Development Council, Leigh has influence on the strategic direction for all products in the Rational brand. Prior to working on Rational brand products, Leigh spent 7 years in the WebSphere development lab as architect for the systems management component of the Application Server. Leigh holds a Masters degree in Computer Engineering from University of Texas at Austin.