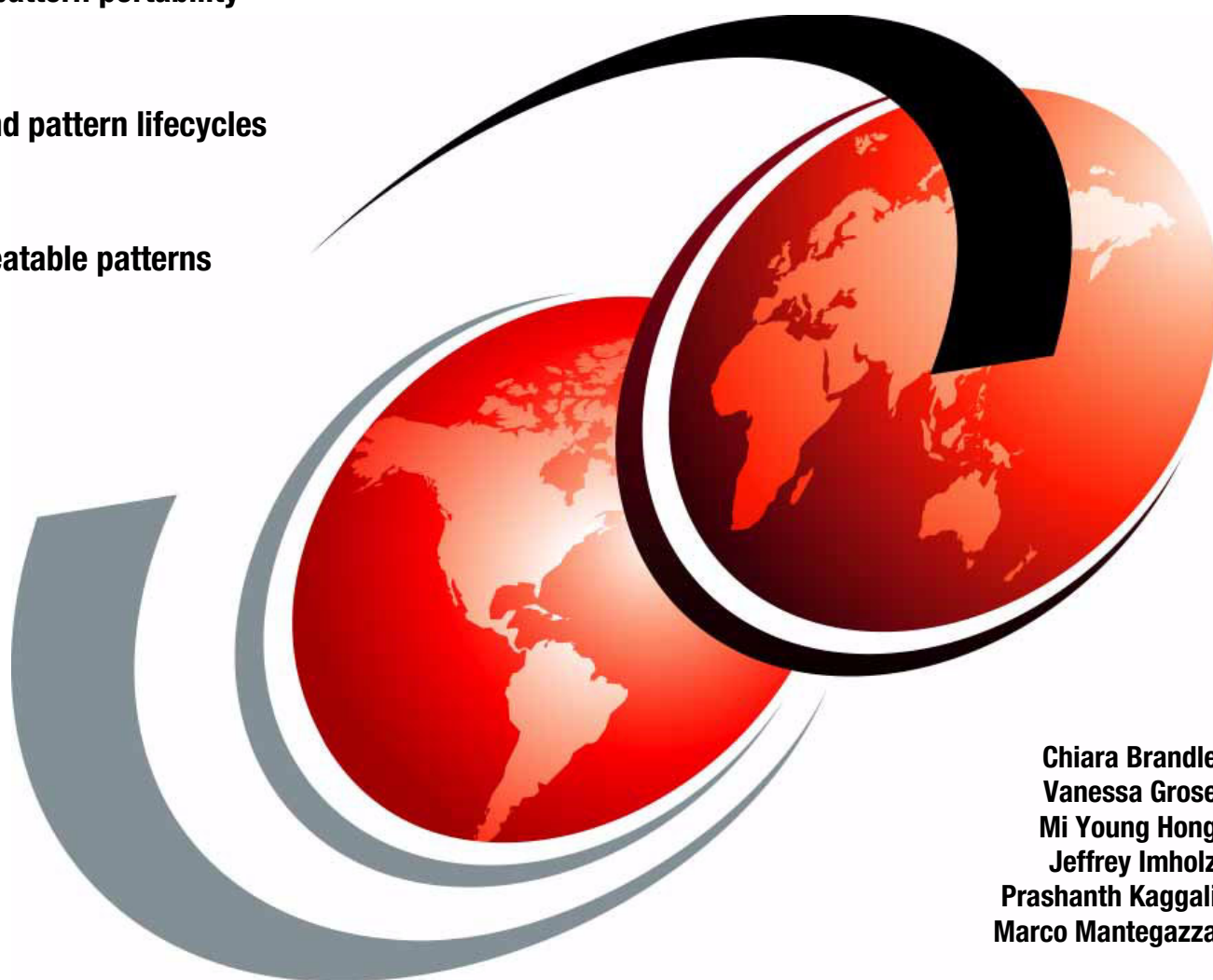


# Cloud Computing Patterns of Expertise

Discover pattern portability

Understand pattern lifecycles

Build repeatable patterns



Chiara Brandle  
Vanessa Grose  
Mi Young Hong  
Jeffrey Imholz  
Prashanth Kaggali  
Marco Mantegazza





International Technical Support Organization

**Cloud Computing Patterns of Expertise**

January 2014

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

#### **First Edition (January 2014)**

This edition applies to IBM Workload Deployer 3.1.0.7, IBM PureApplication System 1.1, IBM SmartCloud Orchestrator 2.2.0.0 and IBM SmartCloud Application Services 1.1.

This document was created or updated on June 18, 2014.

© Copyright International Business Machines Corporation 2014. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Icon Solutions</b> .....	ix
<b>Preface</b> .....	xi
Authors .....	xii
Now you can become a published author, too! .....	xiv
Comments welcome .....	xiv
Stay connected to IBM Redbooks .....	xv
<b>Chapter 1. Introduction</b> .....	1
1.1 What is a pattern of expertise? .....	2
1.1.1 Virtual system patterns .....	3
1.1.2 Virtual application patterns .....	5
1.1.3 Database patterns .....	7
1.1.4 Virtual appliance patterns .....	8
1.2 Tools for customizing patterns .....	8
1.3 Value of patterns .....	9
1.3.1 Simplify and accelerate cloud deployment .....	10
1.3.2 Optimize deployment .....	11
1.3.3 Lower risks and reduce costs .....	11
1.4 Pattern portability .....	11
1.4.1 Available cloud environments .....	12
<b>Chapter 2. Pattern design</b> .....	17
2.1 Pattern usage scenarios .....	18
2.1.1 Pattern sources .....	18
2.1.2 Pattern usage .....	19
2.2 Designing virtual system patterns .....	20
2.2.1 VSP elements .....	20
2.2.2 Tools .....	22
2.2.3 Procedure .....	24
2.2.4 Design example .....	26
2.3 Designing virtual application patterns .....	32
2.3.1 Elements .....	33
2.3.2 Tools .....	36
2.3.3 Procedure .....	38
2.3.4 Design example .....	40
2.4 Design considerations .....	49
2.4.1 VSP versus VAP .....	49
2.4.2 Pattern reuse .....	50
<b>Chapter 3. Pattern deployment</b> .....	53
3.1 Pattern deployment process .....	54
3.2 Sources of Patterns .....	56
3.3 Deployment on IBM SmartCloud Platforms .....	56
3.3.1 IBM Workload Deployer .....	57
3.3.2 SmartCloud Orchestrator .....	66

3.3.3 IBM PureApplication System. . . . .	68
3.3.4 SmartCloud Application Service . . . . .	76
3.4 Deployment solutions made easy with patterns . . . . .	81
<b>Chapter 4. Pattern lifecycle . . . . .</b>	<b>85</b>
4.1 Overview . . . . .	86
4.2 Virtual system lifecycle . . . . .	87
4.2.1 Customizing Hypervisor Edition images . . . . .	88
4.2.2 Lifecycle of a deployed virtual system. . . . .	98
4.3 Virtual application lifecycle . . . . .	105
4.3.1 Base operating system image lifecycle. . . . .	106
4.3.2 Deployed virtual application lifecycle . . . . .	109
4.4 Red Hat OS Update Service . . . . .	120
4.5 Monitoring deployed instances . . . . .	122
4.5.1 Integrating external monitoring systems . . . . .	123
4.5.2 Monitoring databases in IBM cloud technologies . . . . .	125
4.5.3 Using the integrated monitoring capabilities in PureApplication System . . . . .	130
4.6 Pattern management and continuous delivery . . . . .	141
4.6.1 Virtual system development and continuous delivery . . . . .	141
4.6.2 Virtual application development and continuous delivery . . . . .	144
<b>Chapter 5. Pattern portability. . . . .</b>	<b>145</b>
5.1 Standard pattern package formats . . . . .	146
5.2 Import and export prerequisites . . . . .	147
5.2.1 Hardware requirements. . . . .	148
5.2.2 Software requirements . . . . .	148
5.2.3 CLI download and setup . . . . .	148
5.2.4 Using the CLI . . . . .	149
5.3 Exporting and importing patterns . . . . .	150
5.3.1 Working with virtual system patterns. . . . .	151
5.3.2 Working with Virtual Application Patterns . . . . .	156
5.3.3 Working with database patterns . . . . .	162
5.3.4 Working with pattern types . . . . .	164
5.3.5 Working with database workload standards . . . . .	166
5.4 Portability example that uses IBM SCAS and PureApplication System . . . . .	169
5.4.1 Exporting from SCAS . . . . .	169
5.4.2 Preparing PureApplication System to import the VAP . . . . .	169
5.4.3 Importing a VAP into IBM PureApplication System. . . . .	171
5.4.4 Deploying your pattern . . . . .	172
5.5 Leading practices . . . . .	174
5.5.1 Uniqueness. . . . .	174
5.5.2 Size Limits. . . . .	175
5.5.3 Security . . . . .	175
5.5.4 Automation . . . . .	176
<b>Chapter 6. Case studies . . . . .</b>	<b>177</b>
6.1 Enabling insurance on the go by using the IBM Mobile Platform Pattern . . . . .	178
6.1.1 Background. . . . .	178
6.1.2 Solution requirements. . . . .	178
6.1.3 The Open Insurance application. . . . .	179
6.2 Using patterns to cloud-enable a mobile application in the public cloud. . . . .	186
6.2.1 Creating a SmartCloud Application Workload Services instance . . . . .	186
6.2.2 Loading the Mobile Application Platform Pattern into the SmartCloud instance. . . . .	188
6.2.3 Defining environment profile and connecting Worklight Studio to the SmartCloud	

instance .....	189
6.2.4 Generating an Open Insurance pattern from Worklight Studio .....	191
6.2.5 Deploying a pattern in the public cloud .....	193
6.2.6 Rapidly updating a running application instance .....	197
6.2.7 Export pattern content for use in the private cloud .....	200
6.2.8 Public cloud pilot program summary .....	201
6.3 Transitioning to a private cloud for added flexibility and control .....	203
6.3.1 Importing the Open Insurance pattern into PureApplication System .....	203
6.3.2 Making the pattern production-ready .....	205
6.3.3 Deploying the pattern to automatically integrate with PureApplication System's enterprise shared services .....	206
6.3.4 Forcing a virtual machine failure to see automatic failover in action .....	209
6.3.5 Drive load against the application to see scaling in action .....	211
6.3.6 Applying middleware-level fixes to the virtual application instance .....	213
6.3.7 Private cloud pilot project summary .....	217
<b>Related publications .....</b>	<b>219</b>
IBM Redbooks .....	219
Online resources .....	219
Help from IBM .....	221



# Notices

This information was developed for products and services offered in the U.S.A.

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, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** 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 states 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 supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	Informix®	Rational®
CICS®	InfoSphere®	Redbooks®
DB2®	Optim™	Redpaper™
developerWorks®	PowerVM®	Redbooks (logo)  ®
Domino®	PureApplication™	Tivoli®
IBM SmartCloud®	PureFlex™	WebSphere®
IBM®	PureSystems™	z/VM®
IMS™	Rational Team Concert™	

The following terms are trademarks of other companies:

IPAS, and Kenexa device are trademarks or registered trademarks of Kenexa, an IBM Company.

Worklight is trademark or registered trademark of Worklight, an IBM Company.

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

Microsoft, Windows, and the Windows logo are trademarks 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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

# Delivering **PureApplication** pattern expertise

**IBM IMPACT**  
PureApplication Award Winner 2013  
Cloud Innovation Award Winner 2014

To leverage the benefits of PureApplication, you need to have the right pattern in place to meet your business needs. But working out the requirements for your pattern and finding the right partner to help you deliver it can be a challenge.

Icon Solutions is a leading IBM Business Partner when it comes to pattern customisation and implementation. We have a number of patterns already on IBM's **Global Solutions Directory** and have delivered solutions to a variety of customers from across different industries.

**Accelerate your pattern development  
– team with Icon Solutions.**

## Our credentials:

- ▲ Award winning PureApplication Practice
- ▲ Delivered a variety of patterns including internet banking, webMethods, TIBCO, WS Commerce & Sterling Order management and Financial Transaction Manager (FTM)
- ▲ Owners of the European Pure User Group

## How we can help:

- ▲ Pattern customisation and implementation
- ▲ Proof of Concept pattern development
- ▲ Sharing best practices and training
- ▲ Pre-sales workshops
- ▲ Dev-ops enablements

Visit [www.pure-revolution.co.uk](http://www.pure-revolution.co.uk)  
Call **+44 20 7147 9955**



THIS PAGE INTENTIONALLY LEFT BLANK



# Preface

This IBM® Redpaper™ publication explains the business and technical value of emerging patterns of expertise in cloud computing, with specific applicability to IBM PureApplication™ System, IBM Workload Deployer, IBM SmartCloud® Orchestrator, and IBM SmartCloud Application Services. It explains how patterns help companies use the different cloud environments that IBM offers. Also included are some preferred practices for helping to ensure pattern portability.

The pattern-based approach is a response to the need to reduce complexity in IT environments, where various skills are required to design, test, configure, and maintain integrated solutions, including clouds. IT managers spend most of their time maintaining applications and application environments, leaving little time to focus on new business needs or to adopt new technologies. As a result, businesses can lack the agility that is needed to be successful in fast-paced, competitive markets.

Pattern of expertise are designed to deliver the following benefits:

- ▶ Faster time-to-value
- ▶ Reduced costs and resource demands
- ▶ Fewer errors and, therefore, lower risk

Patterns make full use of the unique nature of clouds, both private or public. When they are used in the cloud, patterns allow for the dynamic and efficient use of IT resources to achieve consistent results, even when complex solutions are built. In this way, patterns help save time, money, and resources.

This Redpaper aims to show the value that patterns bring to IT managers and the business as a whole.

## Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



**Chiara Brandle** has worked at IBM since 1999. Early on, she worked on the Tivoli® System Management product suite, architecting and implementing enterprise help desk solutions, outsourcing environments, IT assets inventory solutions, data center monitoring systems, and event management platforms. Later, she moved to the WebSphere® channel technical sales team, focusing on WebSphere Application Server, WebSphere Extended Deployment, and WebSphere Process Management suite. Since 2009, she has worked as an IT Specialist at the IBM Innovation Center in Milan, supporting ISVs and Business Partners in adopting IBM technologies, with a focus on cloud solution-related platforms (WebSphere Workload Deployer, Tivoli Service Automation Manager, IBM Service Delivery Manager, Smart Cloud provisioning) and PureSystem technology (primarily PureFlex™ and PureApplication Systems). In 2012, Chiara was named one of four IBM Cloud Top Guns in Europe.



**Vanessa Grose** has spent her 10 years with IBM focusing on core Java and WebSphere technologies, with roles in development, product quality certification, early customer adoption, and technical sales. She currently works on the worldwide technical sales team for WebSphere Foundation, supporting WebSphere Application Server, Liberty Profile, and Intelligent Management and specializing in cloud and virtualization technologies, such as, IBM Workload Deployer, PureApplication System, and the IBM Patterns of Expertise. In this capacity, Vanessa provides pre-sales support for customers around the world and delivers enablement to IBM technical sales teams to keep them up-to-date on the latest portfolio offerings. Vanessa holds undergraduate degrees in computer science and mathematics from Bethel University in St. Paul, MN, and a master's degree in Computer Science from the University of Minnesota.



**Mi Young Hong** is a senior IT Specialist in the IBM Software Group. She has more than 15 years experience in software design and development as a solution specialist, application architect, and developer, and has worked with customers in the finance industry for many years. Currently, she is working as a Rational® Brand Specialty Architect in Korea, working with customers who are interested in building development and test cloud environments.



**Jeffrey Imholz** is an IT professional with more than 13 years of experience spanning application development, application platforms, application integration, and infrastructure. He currently serves as Senior Architect, Infrastructure & Operations, for Nationwide, a leading U.S. insurance company where he is responsible for infrastructure architecture for middleware technologies and is lead architect for his company's first private cloud offering, which was built around Java workloads. Jeff holds a bachelor's degree in Management Information Systems and Finance from Florida State University and leads a workgroup for the IBM System Z Leadership Council.



**Prashanth Kaggali** has more than 13 years of industry experience in pre-sale and solution architecting. He is a IBM Certified Solution Advisor and Cloud Architect and a Linux and OpenSource enthusiastic. He is currently working as technical consultant on cloud enablement in IBM Innovation Center, helping independent software vendors and Business Partners to build application around the IBM SmartCloud stack.



**Marco Mantegazza** has worked at IBM since 2006 and is currently an IT Specialist with IBM Software Group Italy. He has a master's degree in Telecommunication Engineering from Polytechnic University of Milan. He also earned a second-level master's in Information Technology at the Center of Excellence for Research, Innovation, Education and Industrial Labs partnership (Cefriel) Milan. His areas of expertise include WebSphere Application Server, WebSphere Virtual Enterprise, WebSphere eXtreme Scale, IBM Workload Deployer, and IBM PureApplication System.

## Other contributors

This project was led by **Margaret Ticknor**, who is a Redbooks® Project Leader in the Raleigh Center. She primarily leads projects about WebSphere products and IBM PureApplication System. Before joining the ITSO, Margaret worked as an IT specialist in Endicott, NY. Margaret attended the Computer Science program at State University of New York at Binghamton.

Thanks to the following people for their contributions to this project:

- ▶ Rossella De Gaetano
- ▶ David Henderson
- ▶ Mark J Osteraas
- ▶ Kenneth Rea
- ▶ Andre Tost
- ▶ Dominique Vernier

Thanks to the following people for supporting this project:

- ▶ Ella Buslovich, IBM Redbooks Graphics Editor
- ▶ Shari Deaina, IBM Redbooks IT Support
- ▶ Elise Hines, IBM Redbooks Technical Writer
- ▶ Tamikia Lee, IBM Redbooks Residency Administrator
- ▶ Shawn Tooley, IBM Redbooks Technical Writer

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

<http://www.ibm.com/redbooks/residencies.html>

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

<http://www.ibm.com/redbooks>

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>





# Introduction

Patterns of expertise are proven practices and experiences that were developed during client and partner engagements and captured in a way that can be deployed repeatedly. This chapter introduces patterns of expertise in cloud deployment, and describes the value of patterns not only for IT managers, but for the business as a whole.

The chapter describes what patterns are, the value they offer, and how they can be used across different cloud environments.

This chapter includes the following topics:

- ▶ What is a pattern of expertise?
- ▶ Tools for customizing patterns
- ▶ Value of patterns
- ▶ Pattern portability

## 1.1 What is a pattern of expertise?

A pattern of expertise can be thought of as a recipe that combines all of the knowledge an organization acquired during years of complex infrastructure management tasks for optimizing and automating software deployment. A pattern describes, in a logical way, a repeatable solution that is based on specific sets of virtual images, middleware, applications, and runtime configurations. The result of deploying a pattern is a configured, tuned, and optimized application environment.

The PureSystems™ Centre website lists different patterns that were made available by IBM or IBM Business Partners. PureSystems Centre offers a simple way for PureSystems users to obtain PureSystems optimized content, fixes, updates, and access to expert advice. PureSystems Centre can be found at this website:

<http://www.ibm.com/software/brandcatalog/puresystems/centre/>

If a pattern that reflects your business needs is not available, you can modify or extend an existing pattern to accomplish your goals, or you can create a pattern from scratch.

Patterns of expertise in the realm of cloud computing fall into the following categories:

- ▶ Virtual system patterns
- ▶ Virtual application patterns
- ▶ Database patterns
- ▶ Virtual appliance patterns

After a pattern is deployed, it is called a pattern *instance*. You can deploy many instances of a particular pattern.

Patterns support a set of parameters that are specified at deployment time and influence details of the instance that is deployed. Each pattern instance creates the environment that is described in the pattern, which is customized according to the specified parameters. For example, you can create and test production environments that have the same functionality but use different dedicated hardware resources.

Patterns give users the ability to make different choices that are based on their specific IT needs. The high-level differences between the four types of patterns are shown in Figure 1-1 on page 3.



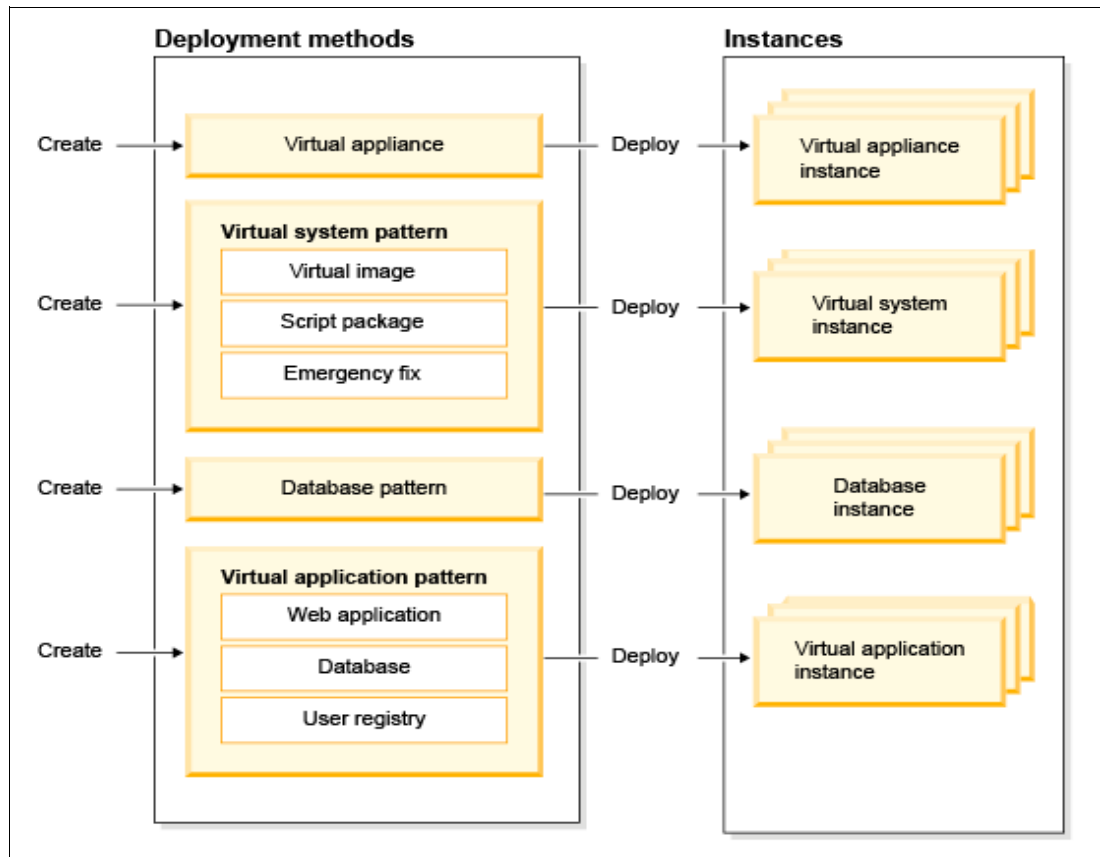


Figure 1-1 Pattern deployment models

### 1.1.1 Virtual system patterns

Virtual system patterns (VSPs) are repeatable topology definitions that are based on various virtual images, each containing multiple middleware components and applications that are configured to work with each other. VSPs provide flexibility and control over the middleware topology that is deployed.

The fundamental building blocks of VSPs are called *parts*. These parts are delivered with the virtual images and are used, with configuration parameters and other artifacts, such as *script packages* and *add-ons*, to create complex VSPs that are deployed as a single unit.

With the push of a button (or by running a client script), you can deploy a single VSP that accomplishes the following objectives:

- ▶ Deploys a topology
- ▶ Provisions a set of VMs
- ▶ Installs an operating system
- ▶ Installs more middleware
- ▶ Configures the environment to host one or more applications

When a virtual system pattern is deployed, the process creates the topology, builds the relationships between the components (for example, federating the custom nodes to the deployment manager), and configures the middleware that is based on the script packages that you provide. The topology can be extensive and contain support for scaling, caching, high availability, and fault-tolerance functions.

Figure 1-2 shows a sample virtual system pattern in which a topology is defined that consists of a WebSphere Deployment manager component, two WebSphere custom nodes, an IBM HTTP Server, and a DB2® HADR Express database (with its primary and secondary components). As you can see, the number of custom nodes and IBM HTTP servers can vary by modifying a parameter.

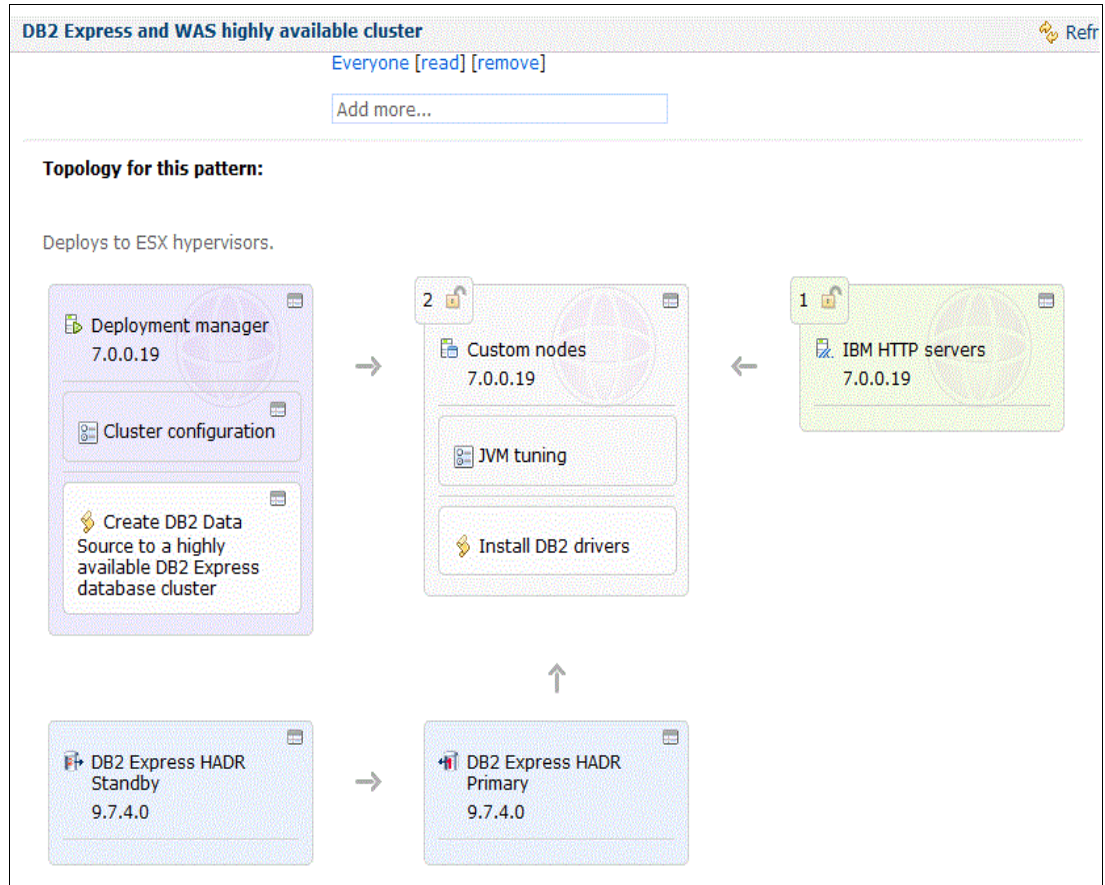


Figure 1-2 Virtual system pattern example that is shown from IBM Workload Deployer

After a VSP is deployed, you can directly access and manage all of the VMs in the delivered topology. This is accomplished by starting a Secure Shell (SSH) connection or by using a Virtual Network Connection (VNC) session. Figure 1-3 on page 5 shows how you can start an SSH session to connect to the wanted system from the Virtual System Instances view.

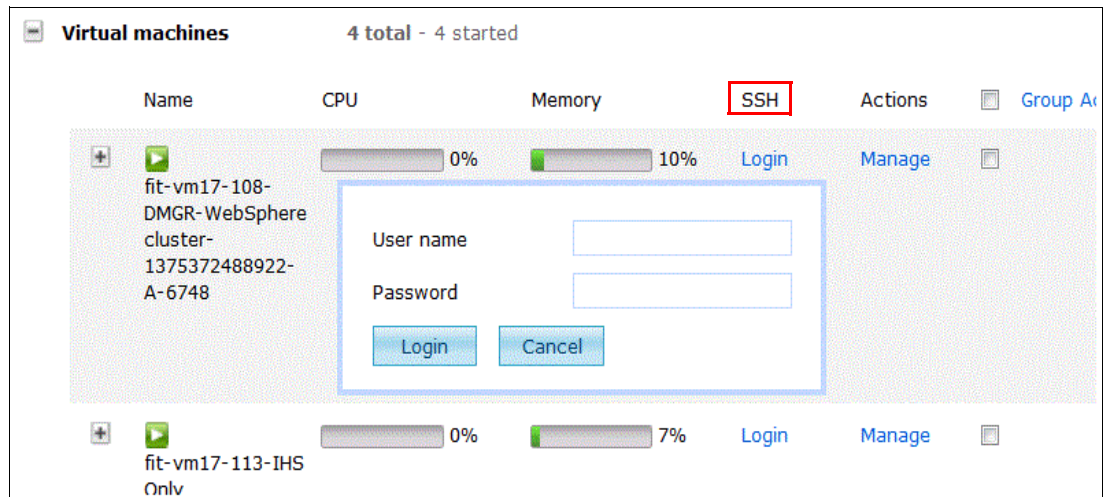


Figure 1-3 Accessing VMs by starting an SSH connection

### 1.1.2 Virtual application patterns

Virtual application patterns (VAPs) take an application-centric approach. A VAP offers a view of a virtual application with which the user can focus only on application requirements and not on the underlying infrastructure that is needed to support the runtime environment. You create the virtual application from the required components, define dependencies between the components, and apply policies to automatically manage the behavior of the application after it is deployed. By using this approach, a user can use the VAP elements (such as components, links, and policies) that do not directly map to VMs.

As an example, Figure 1-4 on page 6 shows the IBM Worklight® virtual application pattern. The pattern consists of the Worklight Server enterprise application, the Worklight Runtime and Worklight Reports databases, the Worklight Configuration component (which is used to configure Worklight Console security), a Worklight Application, and the links that connect all the components.

**Note:** For more information about the IBM Worklight Virtual Application Pattern example, see section 2.3.4, “Design example” on page 40.

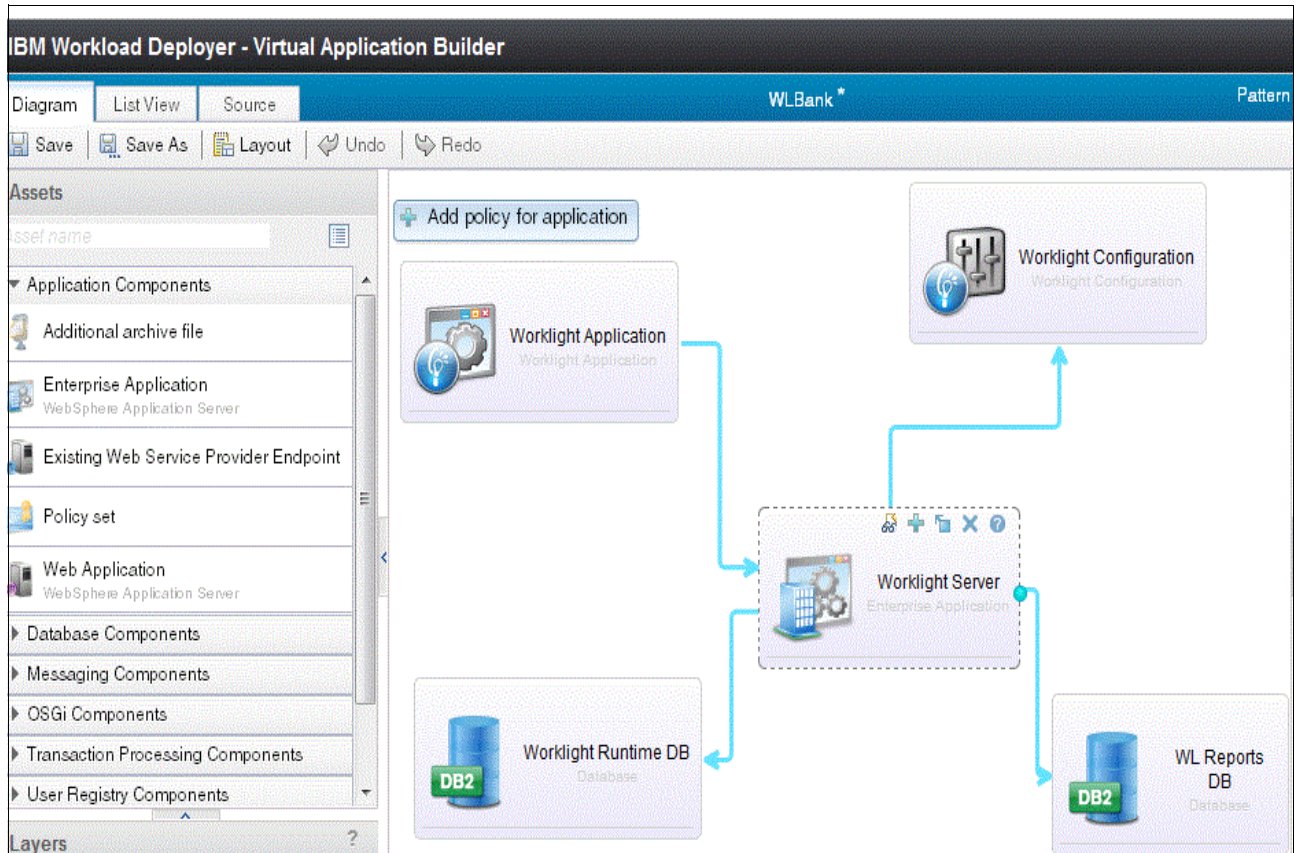


Figure 1-4 Virtual application pattern example that is shown from IBM Workload Deployer

A VAP offers a more abstract, coarser-grained view than a VSP. This means that the number of components that are included in a VAP does not indicate the number of VMs that are deployed. The components in the pattern together can represent functionality that is combined into a single VM, or a single component can lead to the deployment of several different VMs.

Whether it is activated from an application interface or by running a client script, the VAP acts as a template that is converted through a series of transformations into a deployed set of VMs, including all of the middleware and the required configuration. The transformations are handled by plug-ins, which hold all of the information that is needed to turn the abstract VAP into a concrete topology model.

The elements, components, links, and policies that make up a VAP are grouped in *pattern types*. IBM offers various predefined pattern types and predefined patterns that you can use to create your VAP. If your VAP implementation requires any custom content that is not available, you can create your own plug-ins and pattern types by using the Plug-in Development Kit (PDK). The PDK is a compressed package that is available as a download from developerWorks® at the following website. It includes a plug-in and pattern type build environment, samples, and a tool to create a plug-in starter project:

<http://www.ibm.com/developerworks/cloud/library/cl-puresystem-plugintasks/>

The custom content that is needed to create a VAP can be third-party software or an enhancement of existing software that is already available.

The VAP prevents the user from directly accessing and managing the VMs that were created to support the virtual application, so it includes support for managing the lifecycle of the contained artifacts. Scripts and other logic can be used to start and stop all resources that are related to the virtual application, or apply changes to it. This support for lifecycle management is an important difference between VSPs and VAPs. For more information about these differences, see Chapter 4, “Pattern lifecycle” on page 85.

### 1.1.3 Database patterns

Most application scenarios require a database, but databases are distinct entities with their own administrators and lifecycles, and they are often managed independently from the applications that rely on them.

To model this framework as accurately as possible, developers use database patterns. Database patterns are IBM DB2 product extensions that are used to build DB2 databases that are linked to a virtual application as an existing database component. The existing database component can be a database pattern instance that is managed within the cloud environment, or it can be a remote database that was created and is managed outside of the cloud environment.

Database Patterns can create, delete, and update databases that are independent of the virtual application that is using them. But, deleting a virtual application that is using an existing database has no effect on the deployed database or database pattern.

It is also possible to deploy a database as part of a virtual application. In this case, instead of including an existing database in the virtual application pattern, you can include a database component that then becomes a pattern-deployed database service.

In the case of a virtual application pattern-deployed database service, the database is deployed as part of the virtual application. When the virtual application is deleted, the database also is deleted.

When a database pattern is created, you can select the following types of standard workloads to apply to the database instance to be created:

- ▶ Departmental Online Transaction Processing (OLTP)
- ▶ Dynamic Data Mart

The Departmental OLTP standard is the default and is primarily used and optimized for transactional applications. The Dynamic Data Mart standard is primarily used for data warehousing and is optimized for reporting applications.

After you have your database pattern setup, you deploy it and the database instance is provisioned. The provisioning process makes many decisions for you to tailor your database instance to your needs. The provisioned database instance includes standardization, best practices, and workload optimization that corresponds to the workload standard that is selected. After it is provisioned, you have a fully functional, robust database for your IT environment.



### 1.1.4 Virtual appliance patterns

Virtual appliance patterns are VMs that consist of a single-server workload instance with a preconfigured operating system and all of the middleware, applications, and script packages necessary to automatically deploy and configure the application environment. Virtual appliance patterns do not feature the robust management and monitoring features that are available with virtual system and virtual application patterns. Virtual appliance patterns simplify the delivery and operations of an application and require much less installation and configuration than traditional deployment methods of software.

Virtual appliance patterns address key issues that are related to cloud computing, software licensing, and standardization for independent software vendors (ISVs) and software as a service (SaaS) providers. Virtual appliance patterns are useful when a pattern requires a software product from another vendor, or when corporate standards require that all deployed instances of a VM contain a particular operating system or component.

The benefits of adopting virtual appliance patterns include reduced development and distribution costs, accelerated time to market, and a more secure software delivery system. Virtual appliance patterns are considered to be more secure and reliable than traditional software. By copying a file and powering on the virtual appliance, the required parameters for an application are instantly available.

The IBM Virtual Appliance Factory is a self-enablement toolkit that provides automated tools to help developers create virtual appliances in the Open Virtualization Format (OVF). For more information about Virtual Appliance Factory, see this website:

[https://www-304.ibm.com/partnerworld/wps/servlet/ContentHandler/stg\\_com\\_sys\\_virtual\\_appliance\\_factory](https://www-304.ibm.com/partnerworld/wps/servlet/ContentHandler/stg_com_sys_virtual_appliance_factory)

## 1.2 Tools for customizing patterns

After you identify which pattern best fits your infrastructure and application requirements, you then consider any part or component customizations that must be applied to tailor the pattern to your specific needs.

The following tools are available to customize pattern components or parts:

- ▶ The Virtual System Pattern Editor, Virtual Application Builder, and Database Pattern Editor  
All of the IBM cloud technologies that are described in this publication offer these tools, which are built into the console of each technology. You use these tools to create and manage patterns.
- ▶ The IBM Image Construction and Composition Tool (ICCT)  
This tool is available as a separate download and is used to create or customize virtual images within the cloud environment. ICCT can be downloaded as part of the IBM Virtual Appliance Factory toolkit. On IBM PureApplication System, ICCT can be made available as a virtual application.
- ▶ The command-line interface  
The command-line interface (CLI) is available as a separate utility that can be downloaded from the Welcome page in your cloud environment console. It can be used to manage the cloud environment, including virtual images, patterns, deployed instances, and other entities.

- The REST API

Each IBM cloud environment (PureApplication System, IBM Workload Deployer, SmartCloud Orchestrator, and SmartCloud Application Services) exposes a REST API. The REST API is available on the same IP address or host name that is used to access the cloud environment's GUI and the command-line tool. Only a subset of the functionality of the cloud environment is exposed by using a REST API. With the REST API, you can work with certificates and cloud groups, gather diagnostic information, and trace log files. You can also work with hypervisors, IP addresses, and IP groups.

- The Plug-in Development Kit

This tool is available as a separate download. It is designed to help you build your own plug-ins, which are the basic unit of content for creating, deploying, and managing virtual application workloads.

- The Virtual Pattern Kit for Developers (VPDK)

This tool is available as a separate download. It can be used to extend existing patterns or develop custom patterns locally within a test environment or directly on the selected cloud environment.

For more information about using each of these tools, see 2.3.2, “Tools” on page 36.

## 1.3 Value of patterns

The benefits of using patterns vary based on the pattern type you use. For each deployment, you must decide whether you favor the single-server approach of virtual appliance patterns, the middleware infrastructure-centric approach of virtual system patterns, or the application-centric approach of virtual application patterns.

When you deploy software by using virtual appliance patterns, you essentially deploy a virtual image and then manage it as a stand-alone virtual machine. You are not focused on changing the way that you operate or manage the software. Instead, you are focused on maintaining deep control over the virtual image and improving the automation of the software.

When you deploy software by using virtual system patterns, you manage the environment through the available administration consoles. Instead of focusing on changing how you operate or manage the software, you focus on improving the software delivery. Because virtual system patterns are infrastructure-based rather than application-based, virtual system patterns require more knowledge of the middleware to configure deployments and administer the middleware infrastructure. These patterns are a good option when you need more control over the deployment.

With virtual application patterns, you change almost everything about the environment. These patterns are highly optimized and automated solutions, with built-in features to manage high availability and dynamically react to changing conditions (based on policies that you define to accomplish specific business requirements). Virtual application patterns manage the installation, configuration, and integration of all components. Adopting a VAP-based deployment model requires minimal knowledge of the underlying middleware infrastructure. The patterns can be used to deploy multiple instances of the same application. Compared to the use of a virtual appliance pattern or virtual system pattern, a virtual application pattern typically provides the lowest total cost of ownership and shortest time to value (see Figure 1-5 on page 10). However, these patterns offer fewer configuration options for application builders and deployers. As a result, if more control over the configuration is required, you might have to use virtual system patterns or develop your own pattern and associated plug-ins.

Be sure to understand all of your options and make an informed decision that is based on your use case. The different benefits that are associated with Virtual Appliance, Virtual System, and Virtual Appliance pattern types are shown in Figure 1-5 and described next.

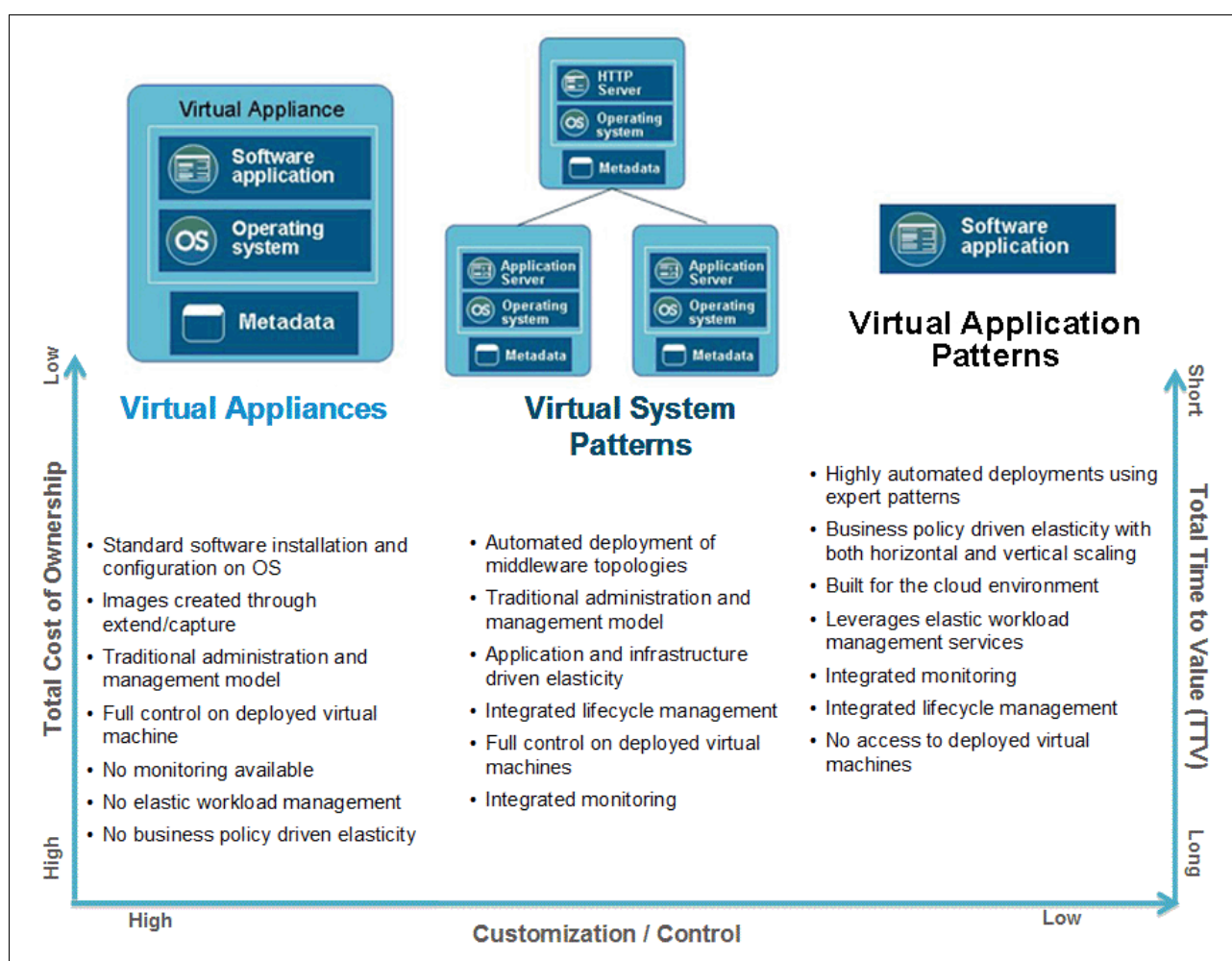


Figure 1-5 Pattern value that is delivered to the organization

### 1.3.1 Simplify and accelerate cloud deployment

The most time-consuming activities for IT departments involve configuring and sizing software, middleware, and virtual system resources to run the applications that are required by the business.

Yet deployment does not have to be a case of constant reinvention. With patterns, you can deploy all needed resources, such as software, middleware, and virtual systems, into your current private cloud infrastructure or a new public cloud environment faster than you might expect.



### 1.3.2 Optimize deployment

By using patterns, the user can specify the wanted outcome (such as required response times or prioritization of multiple applications) to a greater degree than with manual deployments. The company sets its business priorities, and the patterns establish all of the complex parameters that are required to satisfy those priorities.

Patterns of expertise can automatically balance, manage, and optimize all of the involved elements. Users can work with pre-built patterns, have custom patterns that are built for them, or build their own deployable patterns.

### 1.3.3 Lower risks and reduce costs

When applications are deployed through patterns, repeatability is assured and all of the required information that is needed to re-create the environment is retained within the pattern of expertise. Because this information belongs to the business rather than the individual who developed it, it remains with the company. So when employees leave, the skill transfer period is shortened.

Another advantage of using patterns is the savings that are achieved through simplified lifecycle management, which can reduce the cost of maintenance and support because routine tasks are automated.

The cloud environments that are provided by IBM offer different monitoring capabilities for virtual systems that are deployed by using VSPs or VAPs. These monitoring capabilities, which are described in 4.5, “Monitoring deployed instances” on page 122, are made available by patterns and can help not only to reduce administrative costs but also to prevent system outages.

Patterns can also help maintain day-to-day application SLAs and reduce overcapacity by dynamically shifting workloads as needed. Patterns offer policies that administrators or developers can use to set application SLAs and to scale the systems up and down automatically according to the workload. Downtime can be almost avoided by using the high-availability policies that patterns offer. By using well-defined, pre-tested, and pre-integrated patterns that are optimized for the specific needs of the application, you can practically eliminate downtime attributable to human errors that can occur during the installation and configuration stages of creating an application runtime environment.

## 1.4 Pattern portability

The following technologies that are described in this Redpaper use the same technology to support pattern deployment:

- ▶ IBM Workload Deployer
- ▶ IBM SmartCloud Orchestrator
- ▶ IBM PureApplication System
- ▶ SmartCloud Application Services (specifically, IBM SmartCloud Application Workload Services)

This means that pattern portability between these IBM products occurs almost naturally, which gives you the flexibility to choose the technology that best fits your requirements, skills, and budget.

The following constraints should be considered:

- ▶ Virtual system patterns: Because these patterns consist of a set of predefined VMs, script packages, and add-ons, the only constraint for pattern portability is related to the target hypervisor that is supported by the cloud offering.
- ▶ Virtual application patterns: These patterns consist of a set of plug-ins that are responsible for the end-to-end transformation of base operating system images to create the environment that is needed to run the virtual application. However, if the plug-ins are not written in such a way that they can be run on different operating system, VAPs cannot be deployed across multiple cloud offerings, even if they support different types of hypervisors.

If the prerequisite requirements that are declared for your VSPs, VAPs, database patterns, or virtual appliance patterns are met, you can easily reuse your patterns across different cloud environments by exporting them from the source environment and importing them into the target environment.

By using a specific cloud environment that is based on your requirements to deploy your pattern, you can experience different benefits. For example, you can use a pattern to quickly develop and deploy an application in a public cloud environment, and then use the same pattern in a private cloud solution. Alternatively, you can develop and test your pattern in a private cloud and then move it to a high-availability public cloud for production.

You can also use patterns to implement a disaster recovery plan. If there is a failure, patterns can help you quickly redeploy your infrastructure in a different cloud environment.

## 1.4.1 Available cloud environments

This section describes the different IBM cloud offerings, each of which supports pattern-based deployments, and highlights the different functions and hypervisors that are supported by the offerings.

### IBM Workload Deployer

IBM Workload Deployer is a hardware appliance that is packaged with the software that is required to provide support for patterns. By using the appliance, you can manage heterogeneous hardware resources in your cloud environment.

Workload Deployer uses patterns (virtual system patterns, virtual application patterns, database patterns, and virtual appliance patterns) to create consistent, validated, and repeatable applications within a cloud environment. The appliance can deploy applications and topologies (that is, all of the required middleware) into a pool or cloud of virtualized hardware and managed resources.

Workload Deployer supports the following types of hypervisor:

- ▶ IBM PowerVM®
- ▶ IBM z/VM®
- ▶ VMware

By providing these hypervisors to Workload Deployer, the appliance can work with the following topology resources:

- ▶ Cloud groups
- ▶ Networks
- ▶ Storage
- ▶ IP groups

Workload Deployer has the following main features:

- ▶ **Pattern-based cloud delivery**  
The appliance can use virtual system, virtual application, and database patterns.
- ▶ **Monitoring**  
The appliance can monitor all areas of your integrated solution, including virtual system and virtual application instances. The level of monitoring information that is available to each user depends on the role each user is assigned.
- ▶ **Licensing**  
The appliance includes tools for managing software license usage on a per-server basis or by using the processor value unit (PVU) model, depending on the type of license. The appliance tracks licenses as they are used in real time and as virtual machines are created and destroyed.

## **IBM SmartCloud Orchestrator**

IBM SmartCloud Orchestrator integrates the capabilities of several IBM solutions to provide end-to-end service deployment across infrastructure and operating system layers. It provides integrated IT workflow capabilities for process automation and IT governance, resource monitoring, and cost management. It offers an extensible approach to integration with existing environments and facilitates integration with customer-specific service management processes.

SmartCloud Orchestrator can build cloud solutions by using existing IT resources such as hypervisors, networks, and storage.

SmartCloud Orchestrator supports the following types of hypervisor:

- ▶ KVM
- ▶ VMware

SmartCloud Orchestrator includes the following main features:

- ▶ **Pattern-based cloud delivery**  
SmartCloud Orchestrator supports virtual system, virtual application, and database patterns and provides a graphical interface to simplify the composition of cloud automation features.
- ▶ **Designing business processes**  
By using SmartCloud Orchestrator, you can create and edit complex business and IT workflows through simple drag procedures, which can accelerate delivery of IT services.
- ▶ **Self-service user interface**  
SmartCloud Orchestrator provides an intuitive self-service user interface with a customizable catalog of offerings, such as application templates, plug-ins, reusable components, virtual images, virtual patterns, script packages, and add-ons, that are used to build the virtual environment.
- ▶ **OpenStack adoption**  
OpenStack is a cloud operating system that controls large pools of compute, storage, and network resources throughout a data center. Everything is managed through a dashboard that gives administrators overall control while empowering users to provision resources through a web interface. SmartCloud Orchestrator uses OpenStack to define and manage a catalog and repository for virtual disk images, deploy virtual servers on demand, and scale according to the workload. It also provides authentication and authorizations.

- ▶ TOSCA support

SmartCloud Orchestrator supports importing, deploying, and exporting service templates according to the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA). This enables the consumption of standardized third-party content in many heterogeneous cloud environments.

- ▶ Image management

SmartCloud Orchestrator provides image management by using the Image Construction and Composition Tool and Virtual Image Library.

- ▶ Cost management

SmartCloud Orchestrator relies on IBM SmartCloud Cost Management to provide functions for collecting, analyzing, reporting, and billing that are based on usage and the costs of shared computing resources. With this cost management tool, you can track your costs, understand them, and, if needed, invoice them based on allocated or actual resource use by department, user, or some other criteria.

- ▶ Monitoring

SmartCloud Orchestrator supports monitoring workloads and instances by using IBM Tivoli Monitoring.

- ▶ Reporting

SmartCloud Orchestrator provides a diverse set of reports you can use for planning purposes. System usage reports are generated to track physical and virtual resource usage. You can also access reports to track user activity in clouds that are managed by SmartCloud Orchestrator.

For more information about SmartCloud Orchestrator, see 3.3.2, “SmartCloud Orchestrator” on page 66.

## **IBM PureApplication System**

IBM PureApplication System is a workload-optimized and integrated hardware and software solution that is designed to simplify the development, provisioning, and management of consistent, validated, and repeatable applications in a private cloud environment. It features integrated management capabilities that enable self-service provisioning of elastic applications, databases, and middleware.

PureApplication System comes with preinstalled software, including the operating system (Red Hat Enterprise Linux), middleware (IBM WebSphere Application Server), and database (IBM DB2 Enterprise). Related software products are built and optimized for use with PureApplication System but are not included with it.

In PureApplication System, the hardware and software are deeply integrated, which provides a high degree of automation, performance, and simplicity in data center management. The entire system can be managed from single unified interface.

The integrated console in PureApplication System provides management and monitoring interfaces for the following system management functions:

- ▶ System hardware
- ▶ Virtualized storage
- ▶ Networking
- ▶ License usage monitoring
- ▶ User auditing
- ▶ Security configuration

PureApplication System offers the following main features:

- **Pattern-based cloud delivery**

PureApplication System can use virtual system, virtual application, database patterns, and virtual appliance patterns.

- **Monitoring**

PureApplication System includes pre-configured capabilities for monitoring all of the hardware and software components that are provided with the system. The level of monitoring information that is available to each user depends on the role each user is assigned. Facilities for monitoring workloads and instances are accessible from the PureApplication System workload console. Workload monitoring capabilities include hypervisors, operating systems, and entitled middleware and database products.

- **Maintenance**

PureApplication System includes system maintenance features for updating system components (hardware and firmware components, compute nodes, switches, storage, hypervisors, management software, and so on) and workload maintenance features for updating everything that is contained within the virtual machines that run workloads (operating systems, middleware, databases, and updates to virtual system and virtual application patterns). These maintenance functions can be applied independently and are likely to be applied by different teams on different schedules.

#### Licensing

- PureApplication System includes tools for managing software license usage on a per-server basis or by using the processor value unit (PVU) model, depending on the type of license. PureApplication System tracks license use in real time as virtual machines are created and destroyed. In addition to monitoring and optionally enforcing license limits, PureApplication System supports exporting historical license usage information for use in spreadsheets and other tools for analyzing license usage over time.

## **IBM SmartCloud Application Services**

IBM SmartCloud Application Services is IBM's platform-as-a-service (PaaS) offering that enables rapid development and deployment of applications to a cloud with a suite of cloud-based development tools, workload patterns, middleware, and databases. It runs on the IBM self-service public cloud, SmartCloud Enterprise.

IBM SmartCloud Application Services initially includes the following primary services:

- **SmartCloud Application Workload Service**
- **Collaborative Lifecycle Management Service**

SmartCloud Application Workload Service is the PaaS layer in IBM SmartCloud Enterprise and with which you can easily deploy and manage applications by using patterns. You can also customize or create and then deploy virtual system patterns from within this layer. SmartCloud Application Workload Service supports virtual appliance and virtual system, virtual application, and database patterns.

Collaborative Lifecycle Management Service is a set of seamlessly integrated tools that provide a real-time cloud-based collaborative environment for accelerated application development and delivery.





# Pattern design

This chapter describes how to design a pattern in IBM PureApplication System. It explains several pattern usage scenarios that you can choose that are based on your application requirements. It also introduces the overall procedures, building blocks, and tools for designing virtual patterns. Several leading practices for designing patterns of expertise are provided. The chapter also explains, with examples, how to design a pattern by using predefined patterns.

This chapter focuses on designing patterns only in PureApplication System. The remaining chapters in this Redpaper describe how to deploy, import, and export patterns to other cloud environments.

The chapter includes the following topics:

- ▶ Pattern usage scenarios
- ▶ Designing virtual system patterns
- ▶ Designing virtual application patterns
- ▶ Design considerations

## 2.1 Pattern usage scenarios

Patterns of expertise are leading practices and other knowledge that were accumulated by IBM and IBM Business Partners that were captured, tested, and optimized into deployable form. These patterns are not just blueprints or sets of instructions. They are usable, deployable models with the accumulated expertise, such as configuration steps and automated scripts, that are already built into them.

You can use the predefined patterns that are provided with PureApplication System without modification and deploy them to the cloud. You also can create patterns or clone (copy) a pattern and alter its components and configuration to suit the specific needs of your environments.

This section describes some of the typical ways patterns are used for cloud computing.

### 2.1.1 Pattern sources

PureApplication System supports patterns from IBM and IBM Business Partners, and those that you create on your own.

Choose the following best solution that is based on your application and environment requirements:

- Patterns from IBM

IBM includes a number of predefined and pre-optimized virtual patterns for common application topologies with PureApplication System, or the patterns can be obtained separately. These patterns of expertise focus on leading practices and proven topologies for applications that are running on IBM middleware.

- Patterns from IBM Business Partners

IBM Business Partners used their own application and industry expertise to create more patterns of expertise. These partners already produced more than 400 optimized patterns of expertise for various industries.

You can browse and download these IBM and IBM Business Partner patterns of expertise at the PureSystems Centre website:

<https://www-304.ibm.com/software/brandcatalog/puresystems/centre/>

PureSystems Centre offers an easy way for users to obtain PureApplication System-optimized content, fixes, and updates, and to access IBM and IBM Business Partner expertise, including patterns of expertise.

For more information about using PureSystems Centre, see Chapter 4 of *Adopting IBM PureApplication System V1.0*, SG24-8113, which can be found at this website:

<http://www.redbooks.ibm.com/abstracts/sg248113.html?Open>

- Patterns that you create

You can also use your own expertise to create custom patterns of images and components by using tools that are provided by IBM. IBM now provides SmartCloud Workload Service trials with tools for pattern creation, a cloud environment to test them on, and a downloadable Pattern Development Kit (PDK). The SmartCloud Workload Service trial is a 90-day trial that is powered by IBM SmartCloud Services. For more information, see this website:

<http://www.ibm.com/developerworks/cloud/cloudtrial.html>



## 2.1.2 Pattern usage

Patterns enable efficient, repeatable deployment of applications into PureApplication System environments and other IBM cloud technologies, such as IBM Workload Deployer, IBM Smart Cloud Application Services, and IBM SmartCloud Orchestrator. Depending on your requirements, the following options are available:

- Use a predefined pattern

If you find a predefined pattern that meets your requirements, you can use it with no changes. For example, the Business Process Manager Advanced Pattern is a virtual system pattern that provides most of the components that are needed to use the IBM Business Process Manager (BPM) solution in a private cloud. You can use the IBM BPM Advanced pattern without modification to create, deploy, and manage your IBM BPM environments with PureApplication System or Workload Deployer. The pattern helps you accelerate the process of setting up complex, clustered IBM BPM environments in your cloud environment.

- Clone and extend an existing pattern

You might find a predefined pattern that approximately matches your environment but is missing certain elements that you need. In this case, you can clone the existing pattern and then extend it with script packages, add-ons, custom images, and so on.

For example, if you must connect existing Lightweight Directory Access Protocol (LDAP) elements to the IBM BPM solution, or you must configure WebSphere Application Server for your environment, you can extend the existing IBM BPM Advanced pattern by adding a built-in script package that adds your needed features.

- Create a pattern with predefined parts or components

If you cannot find a pattern that fits your environment, you can define a new pattern by using parts and components of other patterns. However, be sure that you are using parts from appropriate pattern types. If you need a virtual application pattern (VAP), you must use elements from another VAP type and not from some other pattern.

For example, if you are going to deploy a web application into a cloud environment by using the Web Application Pattern, you can create your virtual application pattern by adding and connecting the required components in the Web Application Pattern y using a GUI-based application builder.

- Create pattern components

If you plan to use middleware that does not exist in a predefined pattern that is suitable for your environment, you must create your own components. The tools that you use depend on the type of pattern you must create.

If you must create a virtual system pattern (VSP), you can build the necessary virtual images and parts by using the IBM Image Construction and Composition Tool (ICCT) and PureApplication System. If you must create a VAP, you can build the components that you need by using the Plug-in Development Kit (PDK).

For example, if you want to create components for a VAP that supports WebSphere Application Server Community Edition, you must define new configurable application model components for the software and define how to implement (or realize) the deployment of the defined components, and define the lifecycle in the scripts to install, configure, and start software by using the PDK.

## 2.2 Designing virtual system patterns

A VSP is a logical grouping of virtual image parts that together form your middleware topology with associated script packages that further define the environment. VSPs can provide rapid deployment, repeatability, and consistency for simple and complex middleware configurations, while preserving the control and flexibility of traditional middleware environments.

In a VSP, what the administrator defines in the pattern controls is what is deployed in the cloud environment. Figure 2-1 shows how a system topology is defined by using VSPs.

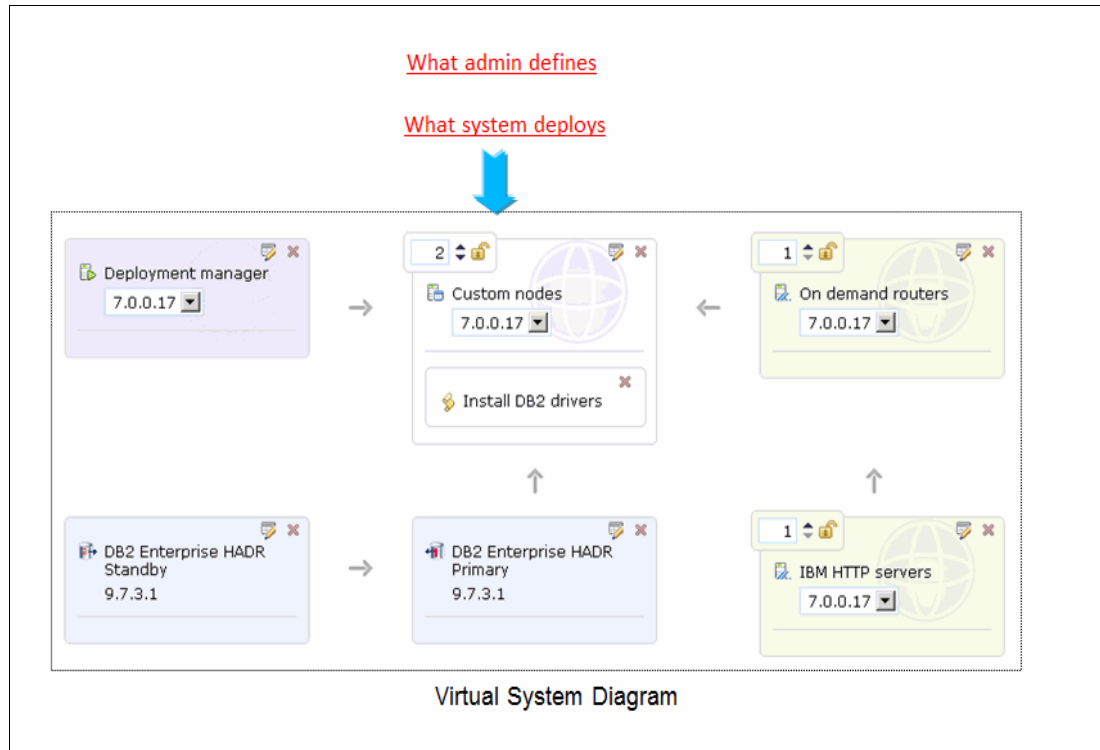


Figure 2-1 Sample system topology that is defined in a VSP

### 2.2.1 VSP elements

VSPs are made up of parts from one or more virtual images, script packages, and add-ons from the PureSystem catalog.

#### Parts

Virtual images contain virtual image parts, which are the main building blocks of VSPs. Each of these parts has properties that you can configure.

The virtual image is a hypervisor edition image that consists of a middleware product, such as WebSphere Application Server that is preinstalled and pre-configured, with an operating system that is designed for virtual environments.

A virtual image includes parts that represent the application server topology components that the image supports. For example, WebSphere Application Server Hypervisor Edition images contain a deployment manager, custom node, and a stand-alone server.

Each part can also contain add-ons and scripts with their own configurable parameters.

Part properties and script parameters can be updated or locked, or both, while you are editing the virtual system pattern that contains them. If the part properties and script parameters are not set during the VSP editing process, the user who is deploying the VSP is prompted for the required property and parameter values.

## Script packages

VSP parts can be customized with script packages. Script packages are flexible and can do almost anything you need, including running **wsadmin** commands or operating system-level commands.

Scripts that automate the installation and configuration of your application are key assets that you can reuse in developing other VSPs.

Script packages are containers that have all of the artifacts that needed to run a script. The script package is a directory that is compressed into a single file that is uploaded to the PureSystem catalog and then associated with VSPs.

## Add-ons

Add-ons are specialized scripts that customize virtual machine (VM) configurations. By using add-ons, you can fine-tune hardware and operating system configurations.

After an add-on is created, it can be dropped onto a topology pattern from the Virtual System Pattern Editor, which is a tool for designing a VSP in PureApplication System. When the pattern is deployed, you provide the parameters for the add-on to use to customize the hardware and operating system configurations.

The following types of add-ons can be added to VSP parts:

- ▶ **Disk:** Adds a virtual disk to the VM and optionally formats the file system and mounts the disk.
- ▶ **NIC:** Adds a virtual network interface controller (NIC) to the VM (NIC add-ons are deployed with environment profiles).
- ▶ **User:** Defines another user on the VM.

**Note:** Although add-ons and script packages have similar configuration steps, the following differences are notable:

- ▶ Script packages have multiple execution-time options that are specified by the user, such as deployment time, delete time, or user-initiated times, but add-ons do not.
- ▶ Add-ons run only at deployment time.
- ▶ You do not specify the order of running add-ons in the Pattern Editor. Add-ons always run before any user-supplied script package.

The add-on definition has a special **TYPE** field that triggers the appropriate hypervisor-level API calls to be started to create the relevant hardware for disk or NIC creation.

## 2.2.2 Tools

PureApplication System provides several tools for customizing VSPs.

### Virtual System Pattern Editor

The Virtual System Pattern Editor is a model-based pattern editor that is integrated into the PureApplication System workload console that you use to manage VSPs. You can edit a virtual system pattern only if it is not read-only and only if you have permission to edit it.

The VSP topology is shown graphically in the Virtual System Pattern Editor. Virtual image parts, add-ons, and script packages can be dragged and dropped onto an editing canvas where you can create or change relationships between the parts.

The Virtual System Pattern Editor provides the following functions that are useful in VSP design:

- ▶ **Creating:** Create a VSP when you need an environment that is not defined by an existing pattern.
- ▶ **Cloning:** Clone an existing VSP and then customize the copy to suit the needs of your environment.
- ▶ **Deleting:** Delete any VSPs that you own.
- ▶ **Modifying:** Modify any VSP that is not read-only.
- ▶ **Setting as read-only:** Make a VSP read-only to prevent further edits to the topology definition and ensure consistent reuse in the cloud.

For more information about how to use the Virtual System Pattern Editor, see the IBM PureApplication System Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.puresystems.appsys.1700.doc/iwd/mpr\\_patediop.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/topic/com.ibm.puresystems.appsys.1700.doc/iwd/mpr_patediop.html)

Figure 2-2 on page 23 shows the Virtual System Pattern Editor in IBM PureApplication System.

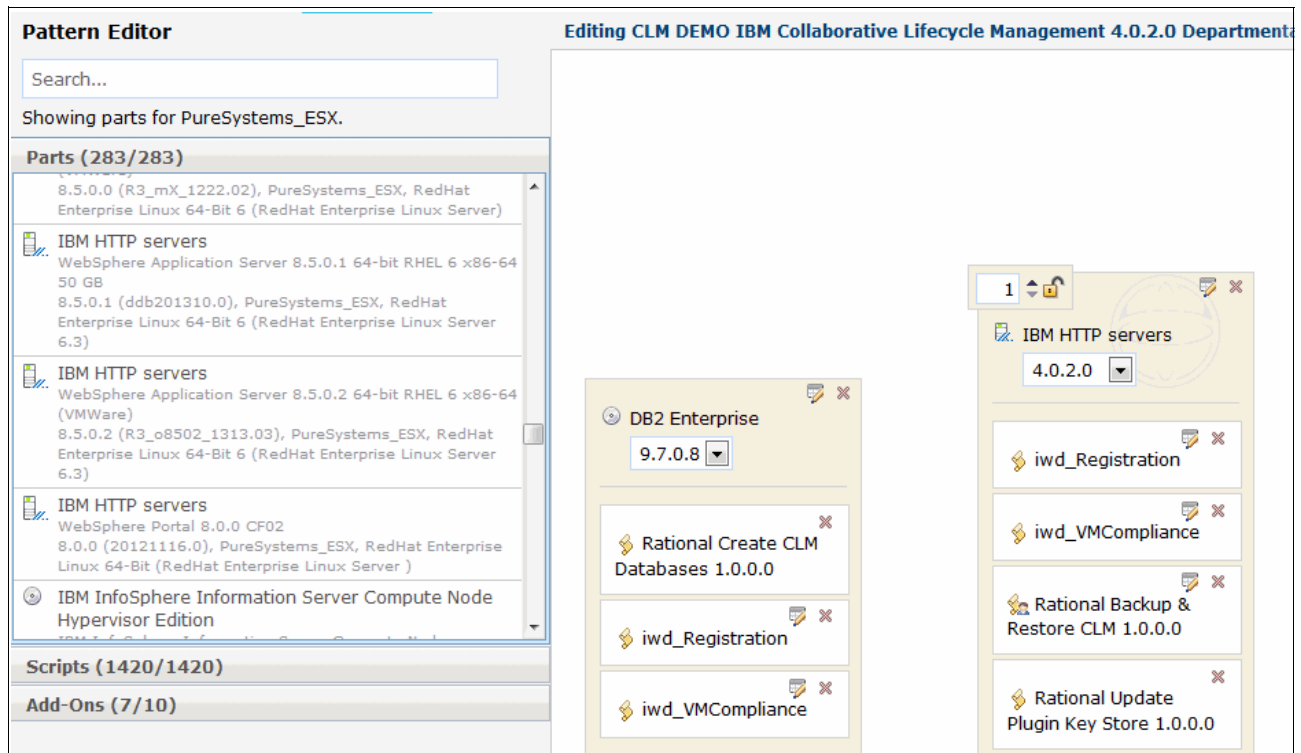


Figure 2-2 Virtual System Pattern Editor

## Image Construction and Composition Tool

The Image Construction and Composition Tool is available as a separate, downloadable tool that you can install and configure to manage the virtual images part of a PureApplication System environment. By using this tool, you can create your own custom virtual images for deployment into cloud environments.

The Image Construction and Composition Tool provides the following functions:

- **Model-driven image creation**

Use the tool to build a model of an image by combining models of a base operating system with software bundles. Each building block contains a semantic and functional model that describes the contents of the components of an image (for example, the products that are installed, supported operating systems, prerequisites, and requirements).

- **Simplified, automated installation process**

After you create the image model, the software installation and image building process is fully automated. It does not require advanced user skills or input regardless of the cloud environment or operating system.

- **Image lifecycle management**

The tool enhances the efficiency of image lifecycle management. Images that you create are self-explanatory; at a glance, you can see the contents of any image. Images that are created by using the tool are also customizable and manageable. You can export images to another system or other users can import your images. Images that are created with the tool have a high degree of interoperability with other IBM products and software.

- Image creation and activation

You can create images of your operating system and software content. Image builders can use the predefined and pre-optimized VAPs to VSPs that were created by experts to build images without requiring their own in-depth knowledge of the operating system or software installation. Specialists build particular software bundles that are then available for image builders to access when images are created.

For more information about how to install and use the Image Construction and Composition Tool, see Chapter 5.3 of *Adopting IBM PureApplication System V1.0*, SG24-8113, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248113.html?Open>

## Advanced Middleware Configuration

Advanced Middleware Configuration for PureApplication System is application release software. It is delivered as a workload that you can use to automate the deployment of existing applications to the cloud (a process that is referred to as *onboarding*). You can also use this software to automate the configuration and deployment of VSPs.

Use Advanced Middleware Configuration when one or more of the following conditions applies to your project:

- You want to deploy applications as virtual system patterns.
- You do not have reliable end-to-end automation for installing and configuring applications.
- Your existing automation is specific to a single topology.
- You want to reduce your investment in low-level automation.
- You want to migrate WebSphere products into the cloud.

For more information about how to use this software, see Chapter 5.3 of *Adopting IBM PureApplication System V1.0*, SG24-8113, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248113.html?Open>

## 2.2.3 Procedure

Depending on your requirements, you can use a predefined VSP as-is, design a new VSP from scratch, or customize an existing VSP to suit your needs.

Figure 2-3 on page 25 shows the VSP design process.

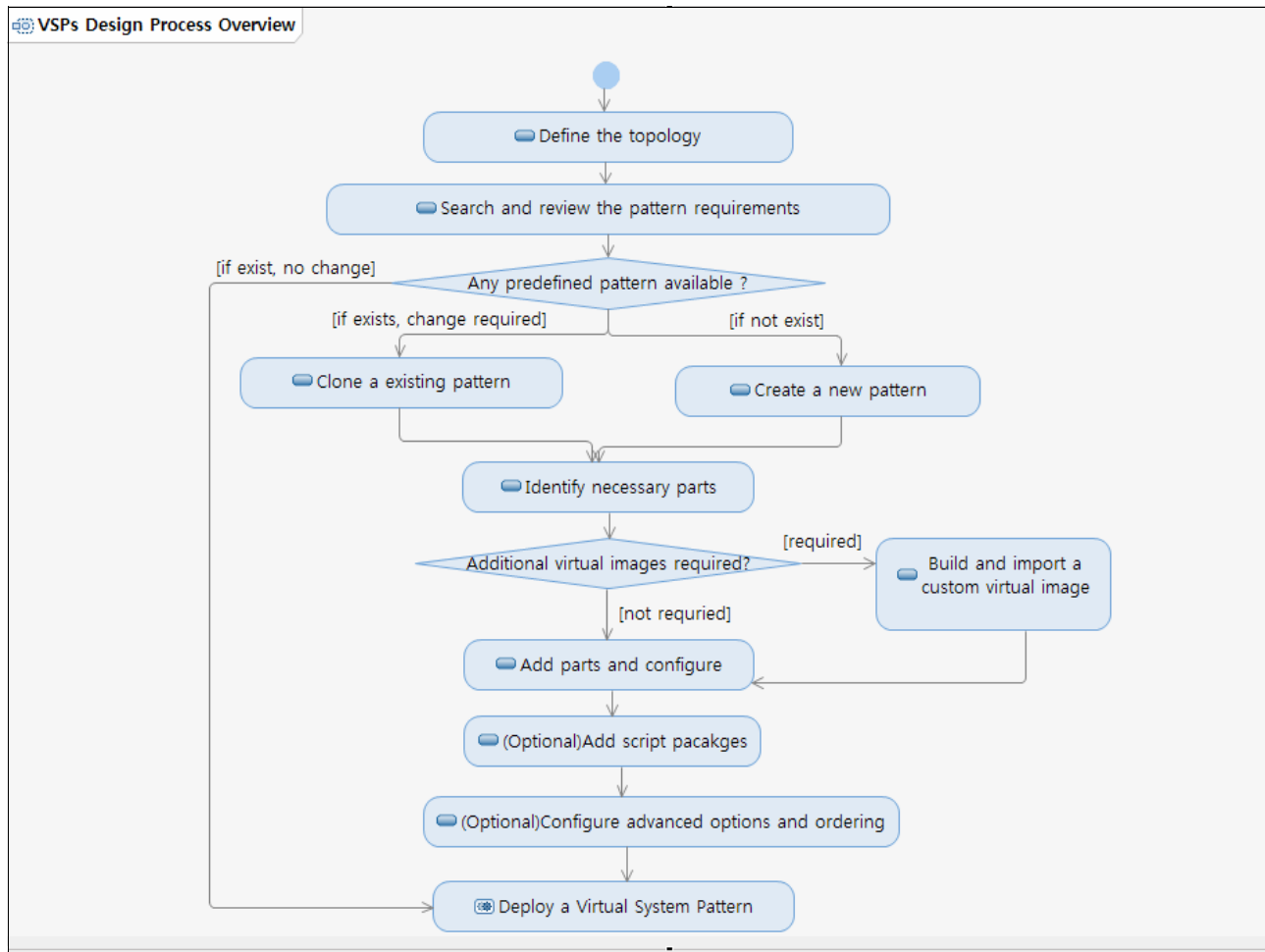


Figure 2-3 VSP design process

The information that is shown in Figure 2-3 can be summarized in the following high-level steps for designing a VSP:

1. Define the topology.

Before you design the pattern, you must analyze your application and define the system topology for your cloud application.

2. Search for patterns and review pattern requirements.

Find patterns that fit your environment by searching for them within resources such as PureSystems Centre. Make sure that you have sufficient resources to run the patterns you select. Typically, a pattern includes documentation that describes the requirements for using it.

3. Choose from the following options that are best for your environment:

- a. Use an existing pattern without alteration (ready for deployment)
- b. Clone and adapt an existing pattern
- a. Create a pattern

4. Identity necessary virtual parts.

You must identity all of the virtual parts of your application and determine whether any other parts must be created with new virtual images.

If required, build and import a custom virtual image.

Virtual images provide the operating system and product binary files that are required to create a virtual system instance.

IBM provides a set of pre-tested virtual images in PureApplication System. However, if you want to build your own virtual image, you can create a custom image by using the extend and capture process or the Image Construction and Composition Tool.

5. Add and configure parts.

After you clone an existing pattern or create a new one, you start modifying the pattern by adding parts from the palette in the Virtual System Pattern Editor and configuring each part's properties for your environment.

6. (Optional) Add script packages.

Script packages are useful in building a custom middleware configuration. The packages can be used to install applications, configure application dependencies, or otherwise tune the middleware layer.

Some example script packages are preinstalled in the PureApplication Systems catalog, but you might have to create your own scripts that are specific to your environments.

7. (Optional) Configure advanced options and deployment order.

You can define the advanced options for VSPs that define your pattern. The options that are available depend on the topology of the virtual system pattern you are editing. These advanced options include configuration settings for patterns such as messaging, session management, and security. You also can change the order in which the virtual image parts are deployed in the topology.

8. Deploy the VSP.

After you design and build your VSP, you can deploy it to your systems.

## 2.2.4 Design example

To illustrate the VSP design process, the authors used the IBM BPM Advanced Pattern. This pattern provides patterns that contain the components that are needed to manage business processes in a private cloud. You can use these virtual system patterns to create, deploy, and manage IBM BPM environments that exist in IBM PureApplication System or IBM Workload Deployer. These pre-optimized patterns help you accelerate the process of setting up complex IBM BPM environments that are highly available.

### Search and review pattern requirements

If you want to set up an IBM BPM environment by using a VSP, find a pattern that is close to what you need for your environment and that you want to clone and review the pattern requirements to ensure that you understand it.

To review the IBM BPM Advanced Pattern requirements and contents, see the IBM Business Process Manager Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.cloud.doc/topics/cbpm\\_priclo\\_gsg.html](http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.cloud.doc/topics/cbpm_priclo_gsg.html)

The following IBM BPM Advanced Pattern packages are available for download:

- ▶ IBM BPM Advanced Pattern on Red Hat Enterprise Linux Server
- ▶ IBM BPM Advanced Pattern on AIX®



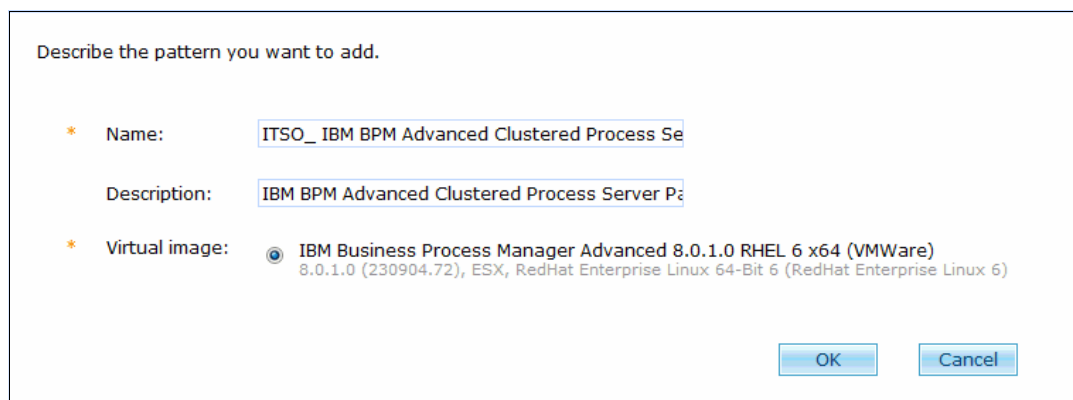
## Clone the pattern

After downloading and installing the pattern, you can start to create the pattern by using the cloning option with predefined patterns.

The following predefined patterns are provided with the IBM BPM Advanced Pattern:

- ▶ IBM BPM Advanced Clustered Pattern
- ▶ IBM BPM Advanced Clustered Process Center Pattern
- ▶ IBM BPM Advanced Clustered Process Server Pattern

Select a predefined pattern to clone and then edit it, including a new name and description that fits your new, cloned pattern, as shown in Figure 2-4.



The screenshot shows a dialog box with the title "Describe the pattern you want to add." It contains three fields, each preceded by an asterisk (\*):

- Name:** The text "ITSO\_ IBM BPM Advanced Clustered Process Se" is entered in the text box.
- Description:** The text "IBM BPM Advanced Clustered Process Server P" is entered in the text box.
- Virtual image:** A radio button is selected next to the text "IBM Business Process Manager Advanced 8.0.1.0 RHEL 6 x64 (VMWare) 8.0.1.0 (230904.72), ESX, RedHat Enterprise Linux 64-Bit 6 (RedHat Enterprise Linux 6)".

At the bottom right of the dialog box are two buttons: "OK" and "Cancel".

Figure 2-4 Giving a cloned pattern a new name and description

## Add and configure parts

When you deploy an IBM BPM pattern, by default it configures the parts of the pattern according to the pre-determined settings that are contained in the pattern. So, you can deploy a pattern without providing any other configuration instructions. However, you always can customize the pattern by adding and removing parts and changing the configuration of those and any other parts in the pattern.

Examples of this process include adding a script package for LDAP configuration, or adding a DB2 database for high availability disaster recovery (HADR). Specifically, you can enable DB2 for HADR by adding and configuring the Process Server standby database part to your pattern, as shown in Figure 2-5 on page 28.

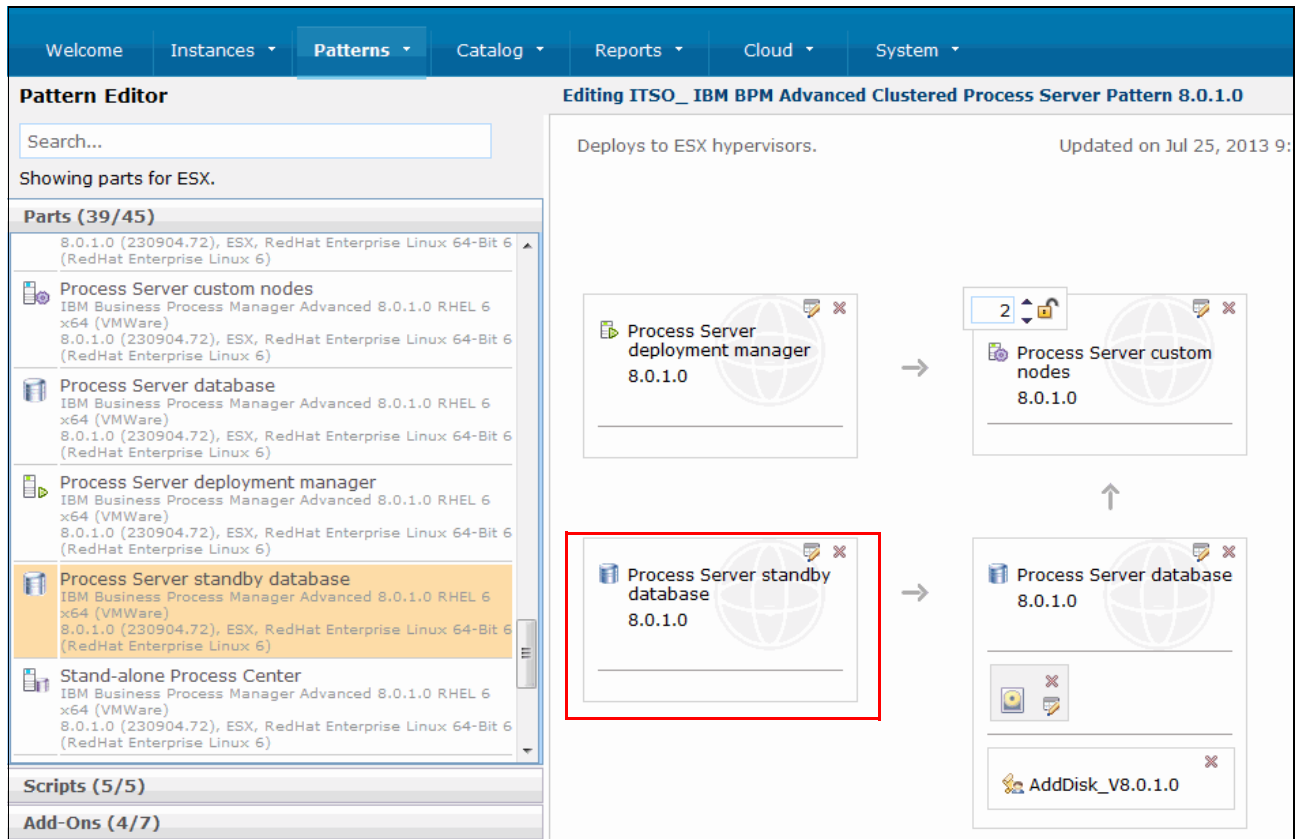


Figure 2-5 Adding a Process Server standby database part to the existing pattern for HADR

## Add appropriate script packages

To customize parts, you can add script packages to VSP topologies. You can also include script packages in VSPs to further define behavior in the IBM BPM Advanced Pattern environment.

The IBM BPM Advanced Pattern packages that are installed on Workload Deployer or PureApplication System include the following script packages that can be used to create customizations:

- ▶ **AddDisk:** Adds disk space to a database virtual machine.
- ▶ **ConfigBPM:** Contains the activation script that configures the IBM BPM environment.
- ▶ **ConfigTDS:** Connects the IBM BPM Process Center deployment manager part or Process Server deployment manager part to an existing LDAP server by using IBM Tivoli Directory Server.

For example, to use the ConfigTDS script package, you must add the package to the Process Server deployment manager as shown in Figure 2-6 on page 29. You can then configure the parameters for your LDAP server in the script package.

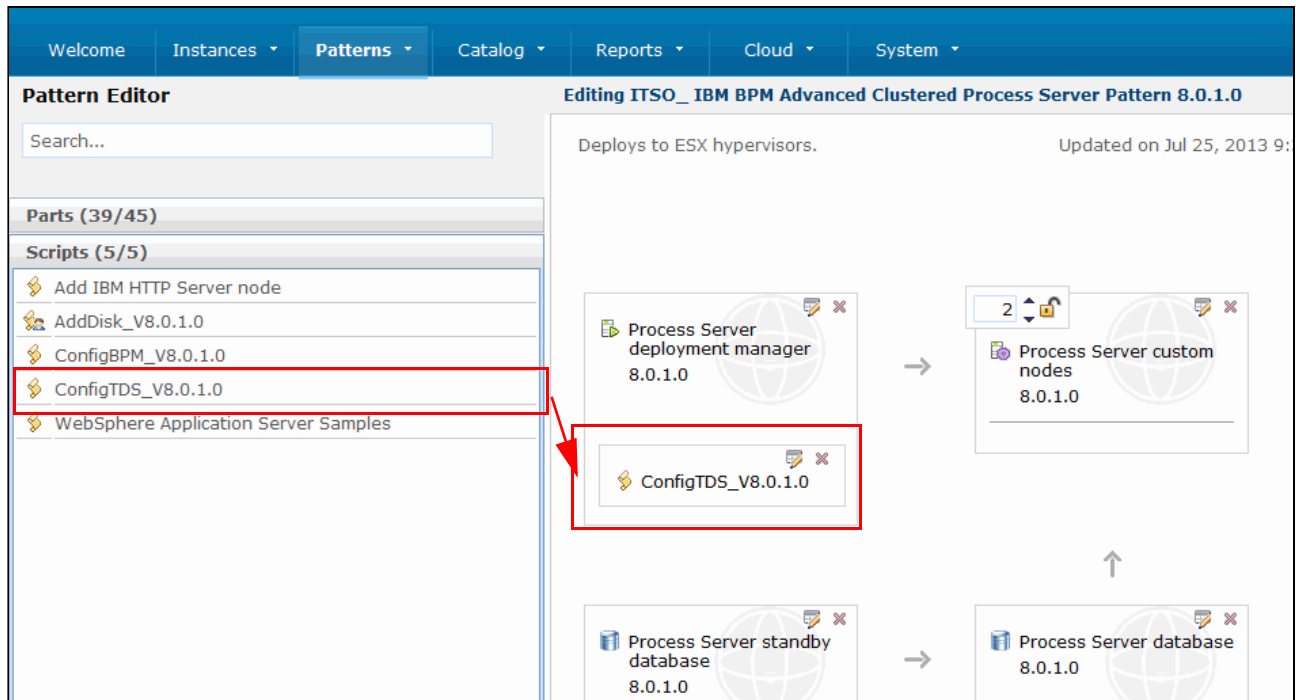


Figure 2-6 Adding a script package to a part for customizing

### Add add-ons to a part

With a VSP, you can include add-ons to a part to customize the pattern. In the predefined Advanced IBM BPM pattern, for example, the default setting for available disk space for the DB2 database part is 30 GB. You can increase this value before deployment by adding a raw disk add-on to the database part.

In this example, you must apply the default raw disk add-on to the database part when you design the pattern, as shown in Figure 2-7 on page 30. If you do not apply the add-on then, only 7 GB of disk space is available on the database VM when the deployment is complete.

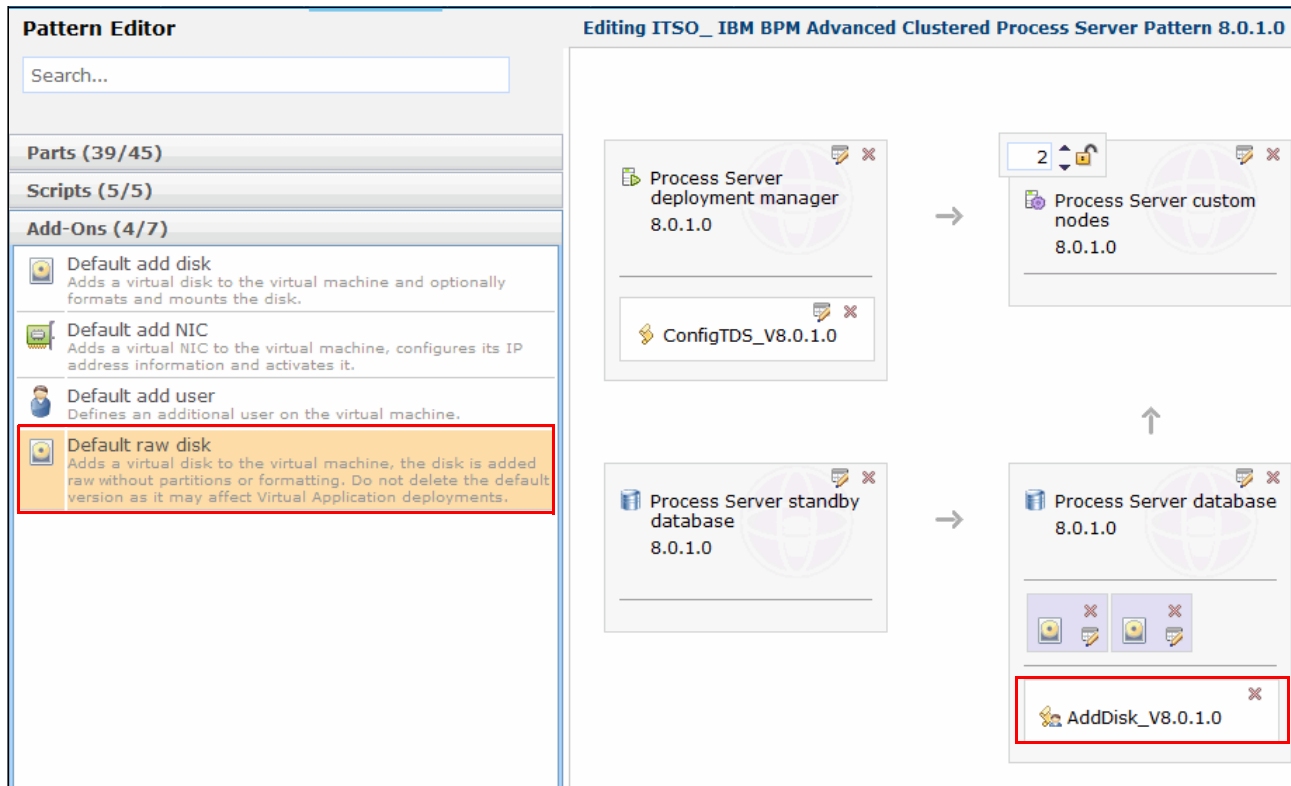


Figure 2-7 Adding a default raw disk add-on to a database part

### Configure advanced options and order of deployment

If you must configure the advanced options for patterns or change the order of deployment of the parts within a pattern, you can make these changes before deployment.

For example, you can increase or decrease the number of IBM Process Server custom node parts and change the order in which some parts are deployed, as shown in Figure 2-8 on page 31.

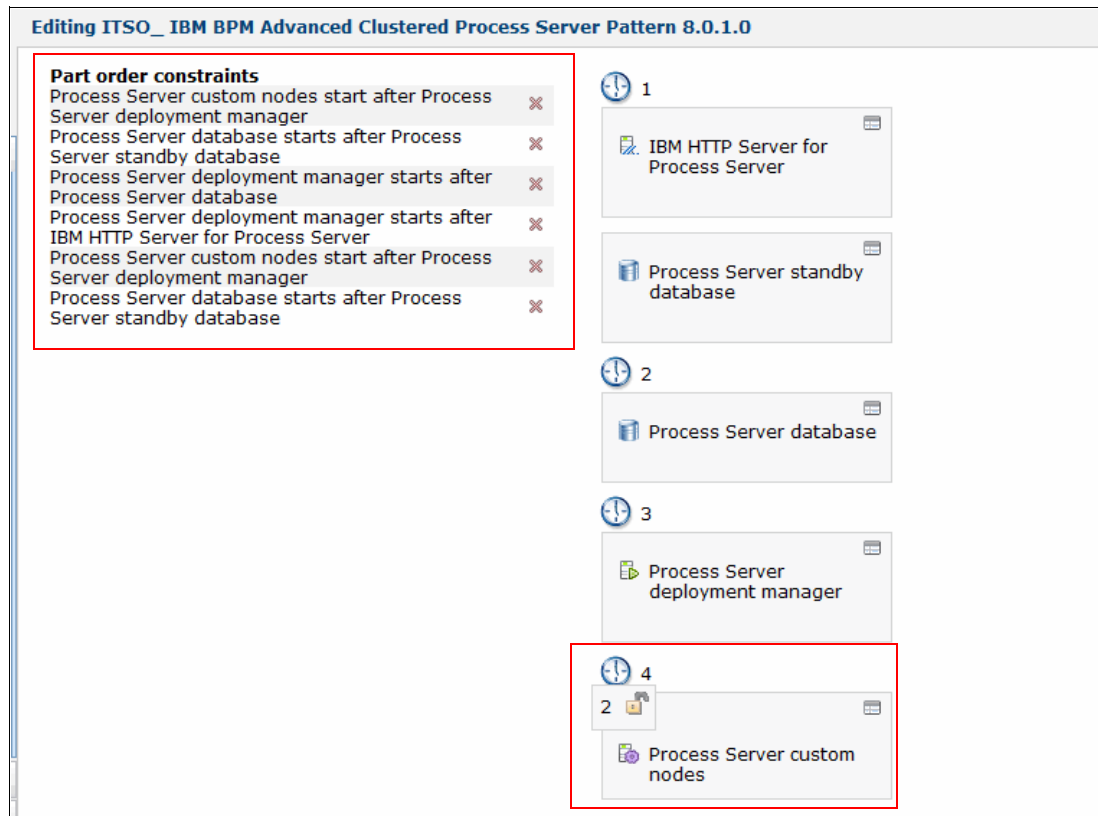


Figure 2-8 Changing the number of process server custom nodes and their deployment order

## Create needed new patterns from scratch

Sometimes you find that it is better to create a pattern by using predefined parts to meet your requirements.

For example, if you need a stand-alone server configuration that is based on the IBM BPM Advanced Pattern for your development or test environments, you can define you own pattern by using the predefined parts. No pattern for the stand-alone server configuration is provided in the IBM BPM Advanced Pattern, but you can design one by using the stand-alone Process Server part and stand-alone Process Center part that are included with the pattern and then connect them as shown in Figure 2-9 on page 32.

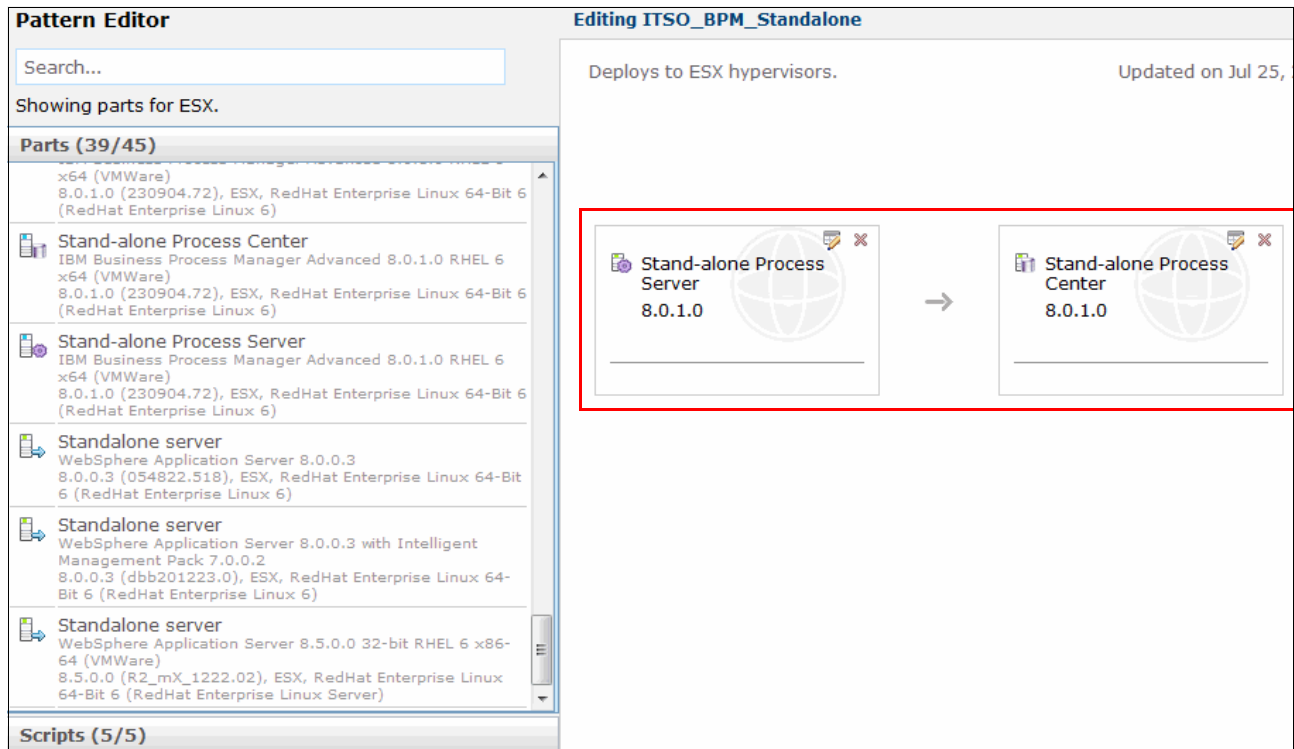


Figure 2-9 Creating a pattern for a IBM BPM stand-alone server configuration

## 2.3 Designing virtual application patterns

Virtual application patterns (VAPs) are a new cloud deployment model that represents an evolution of the traditional topology patterns that are supported in virtual system patterns. Fundamentally, VAPs raise the abstraction layer to a higher point than in virtual system (topology) patterns. VAPs put the focus on the application instead of the application infrastructure.

A virtual application represents a collection of application components and behavioral policies, and the relationships between them. If you define your virtual application pattern in a virtual application model, the system automatically deploys and configures the appropriate middleware components to run your application. This simplifies the deployment process to suit your requirements and monitors the deployment by handling the infrastructure configuration and setup.

Figure 2-10 on page 33 shows the relationship between a virtual application model and the deployed systems.

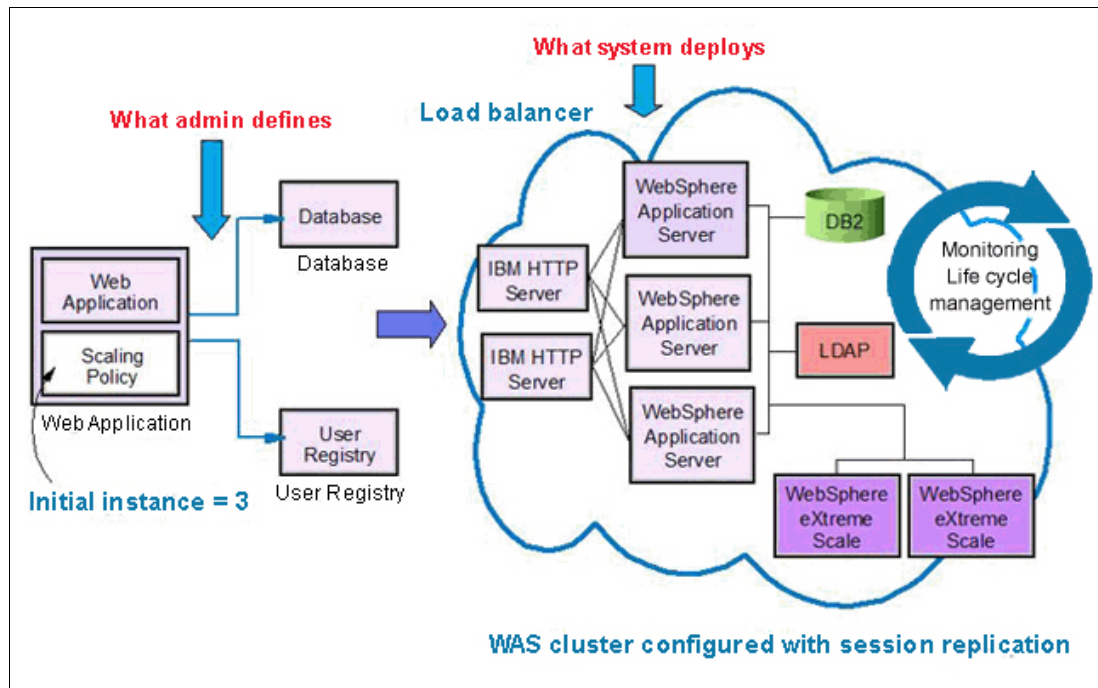


Figure 2-10 Relationship between Virtual application model and deployed system

### 2.3.1 Elements

When you create a VAP, use components, links, policies, pattern types, and plug-ins to create the application model of a virtual application. The components, links, policies, and other configuration options that are available to you are determined by the plug-ins that are included with the selected pattern type.

#### Components

A VAP contains components that represent middleware services that are required by the virtual application instance.

You can connect components in a VAP to indicate dependencies and, optionally, to apply a policy to configure middleware services during deployment to apply specific behaviors or define a quality of service level. As with links and policies, components can include required and optional attributes.

The following components are available with the VAPs that are provided with PureApplication System:

- Application:
  - Additional archive file (Web application)
  - Additional archive file (Java application)
  - Enterprise application component
  - Existing Web Service Provider Endpoint
  - Java application (IBM Java Runtime Version 7)
  - Policy set
  - Web application component

- ▶ Database:
  - Database Studio web console
  - Database (DB2)
  - Existing database (DB2)
  - Existing database (Informix®)
  - Existing database (Oracle)
- ▶ Existing IMS™ database:
  - Messaging
  - Existing messaging service (WebSphere MQ)
  - Topic
  - Queue
- ▶ OSGi:
  - Existing OSGi Bundle Repository (WebSphere Application Server)
  - OSGi Application (WebSphere Application Server)
- ▶ Transaction processing:
  - Existing CICS® Transaction Manager
  - Existing IMS Transaction Manager
- ▶ User registry:
  - Existing user registry (IBM Tivoli Directory Server)
  - Existing user registry (Microstate Active Directory)
  - User registry (Tivoli Directory Server)
- ▶ Other components:
  - Connect out
  - Connect in
  - Connect in (Java application)
  - Connect out (Java application)
  - Monitored file (Java application)

## Links

Links establish communication paths between components. They tell the system that two components must communicate, open the required ports, and configure the middleware for proper communication.

For example, if a web application component depends on a database component, this dependency is defined by the existence of an outgoing link from the web application component to the database component.

## Policies

Policies support the quality of service (QoS) levels of a virtual application. Policies can be applied to an individual component or an entire virtual application.

For example, if you want a web application to be highly available, you can add a scaling policy to the web application component and specify the requirements that trigger scaling to occur, such as passing a processor usage threshold. When deployed, the topology of the virtual application is configured to dynamically scale the web application. Multiple WebSphere Application Server instances are deployed to support the application and other instances are added or removed based on the service levels that are defined in the policy.



The following types of policies can be applied when you build a virtual application pattern:

► **Scaling policy**

Scaling is a runtime capability to automatically expand or reduce your application platform as the load changes. A scaling policy component is used to define this capability (and the conditions under which it occurs) for a specific application.

A scaling policy can be applied to the application hosting environment, including enterprise, web, and OSGi application components. A scaling policy can provide for memory-to-memory session persistence, load balancing, and dynamic scaling of the environment VMs based on workload demand.

Horizontal dynamic scaling adjusts your number of VMs up or down based on certain specified trigger events or measurements. Triggers are grouped according to different application scenarios based on architecture.

The following trigger-related application scenarios are available:

- Static: The elastic scaling feature is disabled.
- CPU-based: Scaling is triggered based on CPU usage only.
- Response-time based: Scaling is triggered based on response time only.
- Web to database: Scaling is triggered based on the number of JDBC connections.

The Static scenario is unique in that it disables elastic scaling. The number of defined instances remains the same no matter what happens regarding workload demand.

► **Routing policy**

You can apply a routing policy to the application component parts of your VAP. By using a routing policy, you can customize the context root for your component. You also can specify the protocol, such as HTTP and HTTPS, for your component.

A routing policy is used with a scaling policy to send requests to virtual applications that are based on unique virtual host names and context roots. In addition, communication between the client and server can be done over SSL (HTTPS).

A routing policy can be applied without having a scaling policy in place. The use of a routing policy requires that the shared proxy is started.

► **Log policy**

You can add a log policy to your application component to specify configurations for log records.

A log policy can be applied to the enterprise OSGi and web application components. This policy defines the server startup trace string level for the application server. If a log policy is configured, it can be modified post deployment from the deployment inlet in the system console.

► **Java virtual machine policy**

A Java virtual machine (JVM) policy controls the characteristics of the JVM.

A JVM policy can be applied to application hosting environments, including enterprise, web, and OSGi components. You can configure a minimum and maximum heap size, enables verbose garbage collection, and enables remote debugging. By using a JVM policy, you can choose between 32-bit or 64-bit versions of WebSphere Application Server.

- **Interim fix policy**

You can apply an interim fix policy to your virtual application to apply updates during deployment and upload emergency fixes to the catalog. When you add an interim fix policy to a virtual application, all applicable fixes for the pattern type and its plug-ins are displayed in the list in the interim fixes URL attribute for the policy. You can select fixes from the list to install during deployment.

Interim fix policies are application-level policies only and cannot be applied to a component. At the time of this writing, only WebSphere Application Server interim fixes for deployments that are based on the IBM Web Application Pattern are supported at the component level.

## **Plug-ins**

A plug-in controls the end-to-end handling of a particular capability in a pattern. When you build a virtual application or database pattern, the plug-in defines the available components, supported linkages and policies, and the configurable properties for each component.

The plug-in also determines the patterns semantics and operation, including the settings that can be configured, how it is deployed, and how it is managed and configured throughout the lifecycle of the virtual application.

## **Pattern types**

A pattern type is a collection of related plug-ins that consist of components, links, and policies for deploying virtual applications and the actions governing their lifecycle. Virtual application domains are defined by pattern types.

For example, the WebSphere Community Edition Sample Pattern Type contains plug-ins. These plug-ins are relative to deployments of simple or complicated web or Java EE applications on the WebSphere Community Server. Deployment depends on certain other features, such as middleware, being installed.

The following virtual application pattern types are provided with the PureApplication System:

- **IBM Foundation Pattern:** Provides shared services, such as monitoring and load balancing, for deployed virtual applications.
- **Application Pattern Type for Java:** Used to build and deploy Java applications.
- **IBM Web Application Pattern:** Used to build and deploy web applications.
- **IBM Database Pattern:** Used to build and deploy database instances.

## **2.3.2 Tools**

The following tools are available for designing and customizing VAPs.

### **Virtual Application Builder**

To create or update a virtual application pattern, use the Virtual Application Builder (VAB) to define the application model. By using the VAB, you can design your application model and graphically assemble and configure your virtual application.

The virtual application model is defined to include certain components, links, and policies that are available within the selected pattern type and any secondary pattern types that are associated with the selected one.

Components are the fundamental building blocks that are used to construct a virtual application model. You start by selecting components on the VAB canvas and configure values for the component attributes. You then can add policies and links between components and configure values for their attributes, as shown in Figure 2-11.

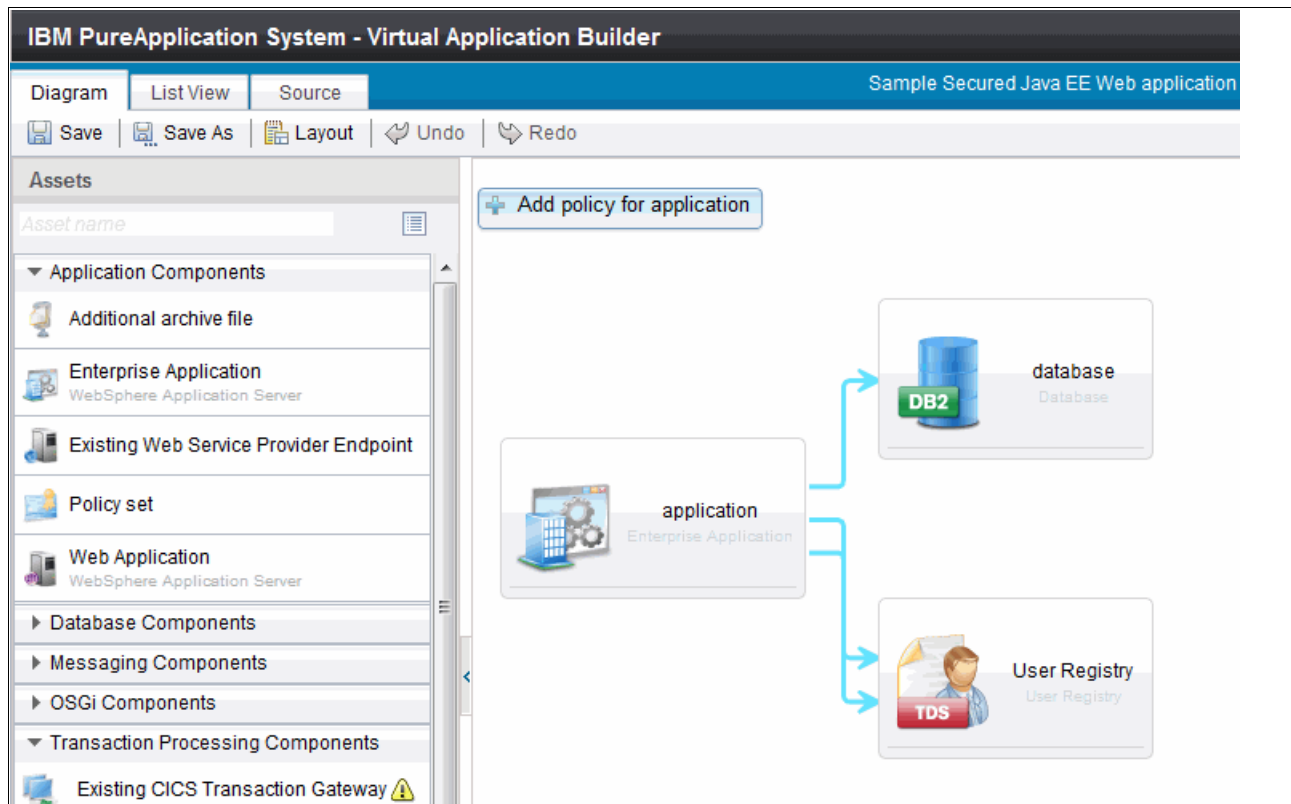


Figure 2-11 Virtual Application Builder

## IBM Workload Plug-in Development Kit

By using the IBM Workload Plug-in Development Kit (PDK), you can create your own custom content that you can add to a VAP by creating plug-ins and pattern types. The custom content can be third-party software or an enhancement to existing software that is already available as a VAP.

The PDK includes an Eclipse plug-in that you can use in your Eclipse or Rational Application Developer environment to create and edit the configuration files in the plug-ins that are used in your virtual applications.

For more information about installing the PDK and using it to design and implement custom plug-ins, see the following resources:

- ▶ Chapter 4 in *Adopting IBM PureApplication System V1.0*, SG24-8113, which is available at this website:  
<http://www.redbooks.ibm.com/abstracts/sg248113.html?Open>
- ▶ *Creating Composite Application Pattern Models for IBM PureApplication System*, SG24-8146, which is available at this website:  
<http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248146.html?Open>

## Virtual Pattern Kit for Developers

The Virtual Pattern Kit for Developers (VPDK) is a self-contained resource that is delivered as a VMware image (to be installed locally) and provides basic pattern development capabilities for developers.

The VPDK includes tools for pattern development, such as the PDK and the following basic patterns:

- ▶ Web Application Pattern 2.0
- ▶ IBM Transactional Database Pattern 1.1
- ▶ IBM Data Mart Pattern 1.1
- ▶ Plug-in Development Kit (PDK)
- ▶ IBM Image Construction and Composition Tool
- ▶ Base OS Red HAt Enterprise Linux (RHEL) image

By using the VPDK, pattern and plug-in developers can gain experience working with PureApplication System, extend existing patterns, or create patterns. You can download the VPDK from IBM developerWorks at this website:

<https://www.ibm.com/developerworks/downloads/ibm/virtualpatternkit/>

For more information about how to install and configure the VPDK tool, see the video that is available at this website:

<http://www.youtube.com/watch?v=nTpk-55kBTc&feature=youtu.be>

### 2.3.3 Procedure

You have two options for designing a virtual application to meet your requirements: reuse pre-existing patterns and configure them, or create a VAP of your own.

VAPs are simple to create, deploy, and monitor. VAPs are the quickest way to deploy applications without worrying about the infrastructure configuration. PureApplication System handles deploying applications for you, which reduces much of the complexity.

The process for designing VAPs is shown in Figure 2-12 on page 39 and listed in the steps that follow the figure.

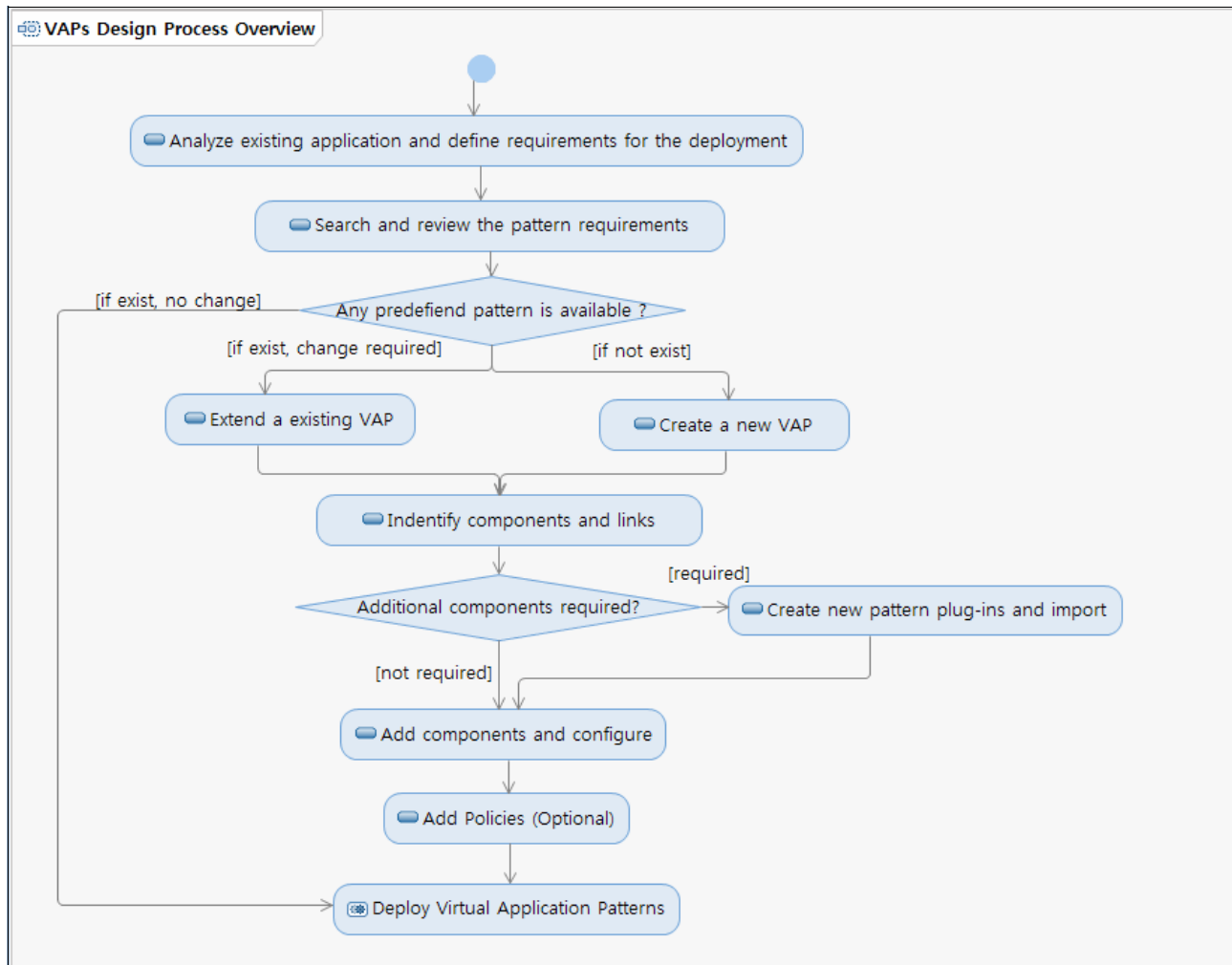


Figure 2-12 VAP design process

The process that is used to design a VAP includes the following main steps:

1. Analyze the existing application and define the requirements for deployment.

Analyze the existing application and identify the application artifacts and other information that are needed to deploy the application to the cloud. Be sure to define the necessary QoS settings for the application, such as any scaling, logging, and monitoring requirements.

2. Search and review the application pattern requirements.

Browse and review the VAPs that are provided by searching for them within resources, such as PureSystems Centre, and determine whether one exists that best fits your needs. If a predefined VAP suits your environment, you can use the pattern as is at no change or extend the pattern and make changes.

If a predefined VAP does not exist to suit your environment, create a VAP.

3. Identify components and links.

Determine which components are required for your application environment. You also should determine whether you can use all components in the predefined pattern type, or whether you must create components.

4. Create and import a pattern plug-in.

If you must add custom components that do not exist in the predefined pattern type that you selected, you must create virtual application plug-ins and pattern types with the PDK.

5. Add and configure components.

Here, you build the pattern with the necessary components and links between the components. The list of available components depends on the template you choose.

Each component has its own properties that can be configured during pattern design. Some properties are required and others are optional.

6. Add QoS policies.

You can apply policies (scaling, logging, monitoring, and so on) to a virtual application or components to establish any specific behaviors that are needed to meet your promised service levels in the deployed virtual application instance.

## 2.3.4 Design example

To illustrate the VAP design process at a high level, the authors used the IBM Mobile Application Platform Pattern type as the basis of their example. This pattern type consists of the IBM Web Application Pattern and the IBM Mobile Application Platform Pattern, which provides a number of components for a mobile environment that is based on IBM Worklight.

### Review the pattern requirement

By using a predefined pattern, you can review and understand the pattern requirements and contents.

Most providers, including IBM, provide a document that explains these details. To review the requirements and the contents of the Mobile Application Platform Pattern, see the IBM Worklight Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c\\_pureapp\\_work\\_with\\_mobile\\_application\\_platform\\_pattern\\_type.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c_pureapp_work_with_mobile_application_platform_pattern_type.html)

### Install the pattern type

If you are using the IBM PureApplication System, you use the PureApplication System Workload Console to install the IBM Mobile Application Platform Pattern Type. For more information about how to install this pattern, see the IBM Worklight Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c\\_pureapp\\_installation.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c_pureapp_installation.html)

### Create a pattern

After the pattern type is installed, you can start designing the pattern by using predefined patterns.

The Mobile Application Platform Pattern Type consists of the Web Application Pattern and the Mobile Application Platform Pattern.

The Mobile Application Platform Pattern provides the following components:

- ▶ Worklight application components
- ▶ Worklight adapter components
- ▶ Worklight configuration components

- ▶ Worklight application component links to enterprise application components (WebSphere Application Server)
- ▶ Worklight adapter component links to enterprise applications components
- ▶ Enterprise application component links to Worklight configuration components
- ▶ Worklight configuration links the user registry (Tivoli Directory Server)

The IBM Mobile Application Platform Pattern does not provide pattern templates; therefore, you must use the provided components to create a pattern in Virtual Application Builder. You start this process by selecting the Mobile Application Pattern Type, then click **Start Building**, as shown in Figure 2-13. You then build your new pattern.

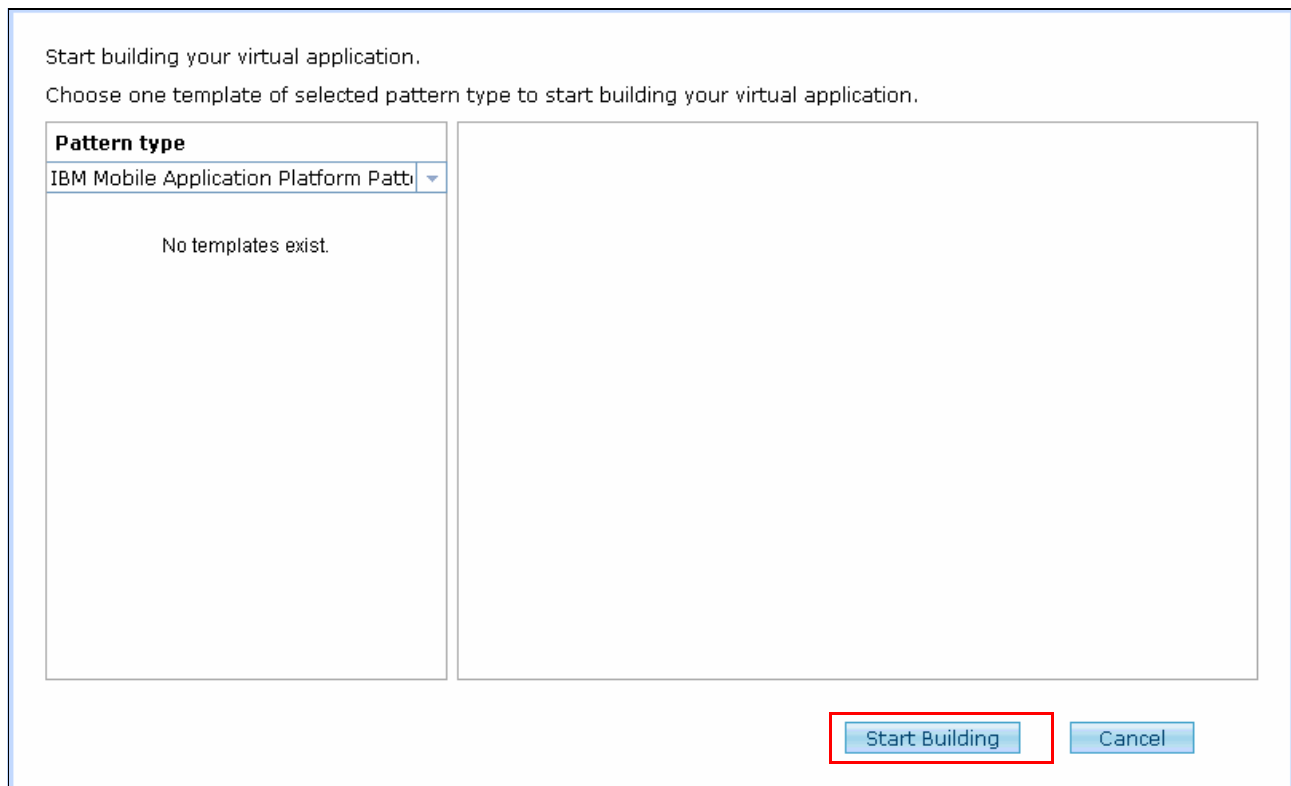


Figure 2-13 Creating a pattern with the Mobile Application Platform Pattern type

## Adding and configuring the pattern components and links

After your pattern is created, you can edit it by adding provided components.

### Creating a Worklight server

Create a Worklight server by using the enterprise application server component in the application component category. In the Virtual Application Builder, components are grouped into categories, such as application, database, messaging, OSGi, transaction processing, and user registry.

In IBM PureApplication System, click the Diagram tab in the Virtual Application Builder. Complete the following steps:

1. From the Assets list, expand Application Components, and then drag an Enterprise Application WebSphere Application Server component onto the canvas.
2. Configure the server properties and supply the information that is shown in Table 2-1 on page 42.

Table 2-1 Worklight server component properties

Property	Description
Name	Name of the Worklight server (for example, WorklightServer).
EAR file	IBM Worklight EAR file that contains the Worklight server package to be uploaded (for example, worklightstarter.ear).

**Note:** You can generate the required EAR file for the IBM Worklight virtual application from your Worklight application workspace by using the Worklight Studio or Ant tasks that are included in the pattern. For more information about how to build an IBM Worklight virtual application, see the product information center at this website:

[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c\\_pureapp\\_building\\_deploying\\_WL\\_VA\\_CLI.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c_pureapp_building_deploying_WL_VA_CLI.html)

Figure 2-14 shows a VAB window view of creating and configuring a Worklight server.

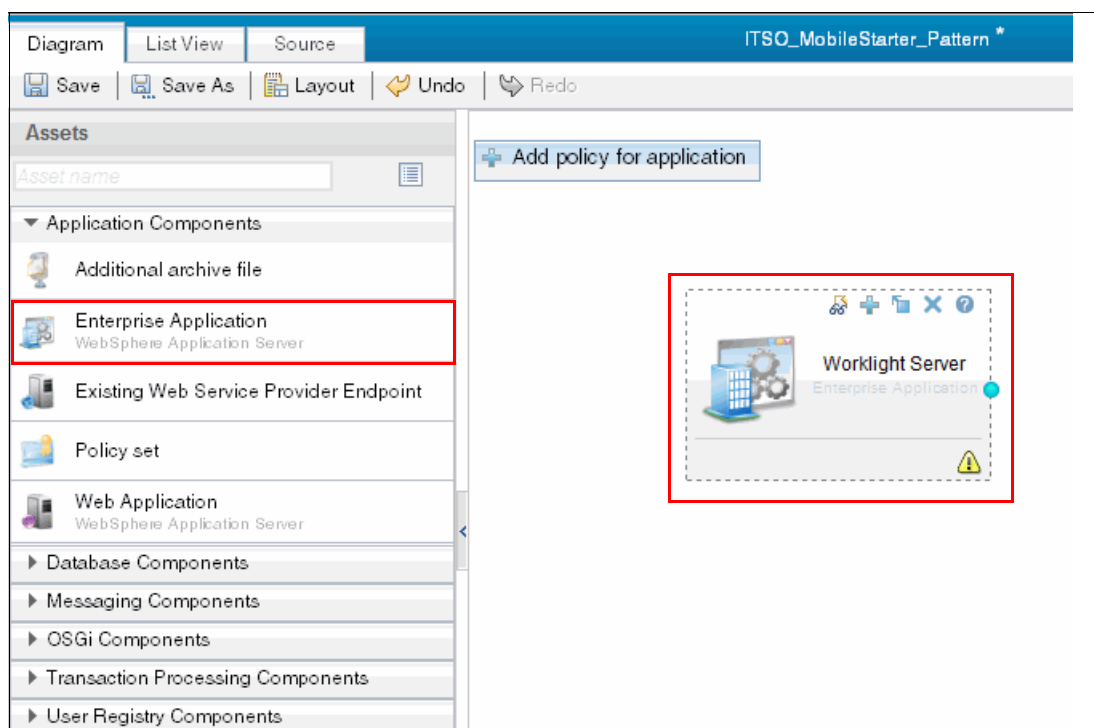


Figure 2-14 Creating and configuring a Worklight server

### Creating Worklight runtime and reports databases

Create two Worklight databases by using the database component. One is for the runtime database and the other is for the reports database.

From the Assets list, expand Database Components, and then drag a Database DB2 component onto the canvas. Use the properties that are shown in Table 2-2 on page 43.



Table 2-2 Worklight database component properties

Component	Property	Description
Runtime database	Name	Name of the Worklight runtime database component (for example, WLRuntimeDB)
	Database name	Name of the runtime database (for example, WLRTIME)
	Source	From the Source list, select <b>Apply a database workload standard</b> , and then click the database workload standard created. For more information, see this website: <a href="http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.worklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html">http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.worklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html</a>
Reports database	Name	Name of the Worklight runtime database component (for example, WLReportsDB)
	Database name	Name of the runtime database (for example, WLRTPT)
	Source	From the Source list, select <b>Apply a database workload standard</b> , and then click the database workload standard created. For more information, see this website: <a href="http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.worklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html">http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.worklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html</a>

**Note:** You must install custom workload standards into your catalog for the Worklight runtime and reports databases before you create the pattern. For more information about installing custom IBM Worklight database workload standards, see the IBM Worklight Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/t\\_pureapp\\_installing\\_DB\\_workload\\_standards.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/t_pureapp_installing_DB_workload_standards.html)

Figure 2-15 on page 44 shows the Virtual Application builder window for creating a Worklight runtime and reports database.

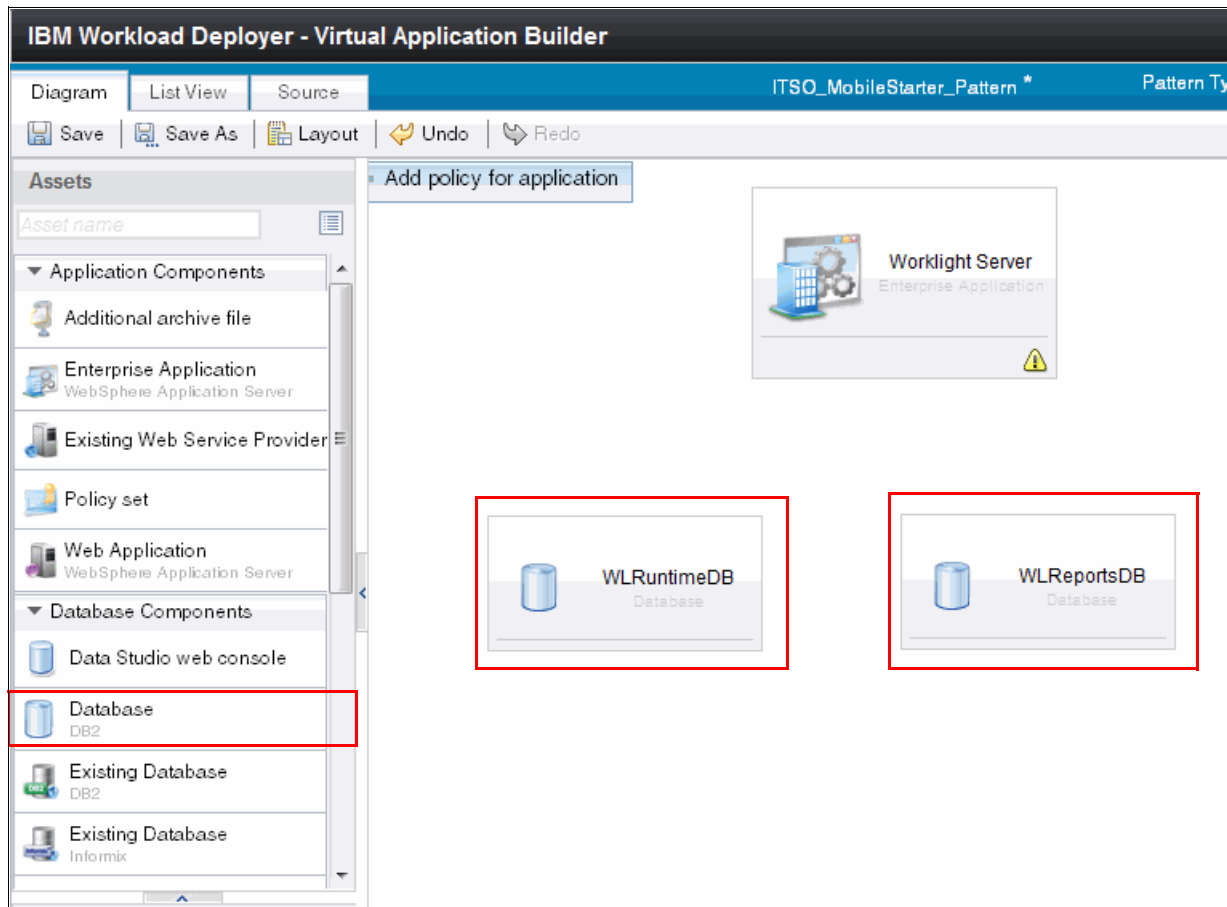


Figure 2-15 VAB window for creating database components

### **Configuring connectivity for the runtime and reports databases**

Complete the following steps to create links from the Worklight server component to the runtime and reports database components:

1. Drag a connection from the Worklight Server component to the runtime database component.
1. In the Resource References of Data Source field, select **jdbc/WorklightDS** for the IBM Worklight runtime database.
2. Drag a connection from the Worklight Server component to the reports database component.
3. In the Resource References of Data Source field, select **jdbc/WorklightReportsDS** for the IBM Worklight reports database. Configure them as shown in Table 2-3.

Table 2-3 Worklight link component properties

Link property	Description
Resource References of Data Source field select jdbc/WorklightDS for the Worklight runtime database	Connection from the Worklight Server component to the runtime database component.
Resource References of Data Source field select jdbc/WorklightReportsDS for the Worklight reports database	Connection from the Worklight Server component to the reports database component.

Figure 2-16 shows the VAB window for configuring connectivity for the Worklight runtime and reports databases.

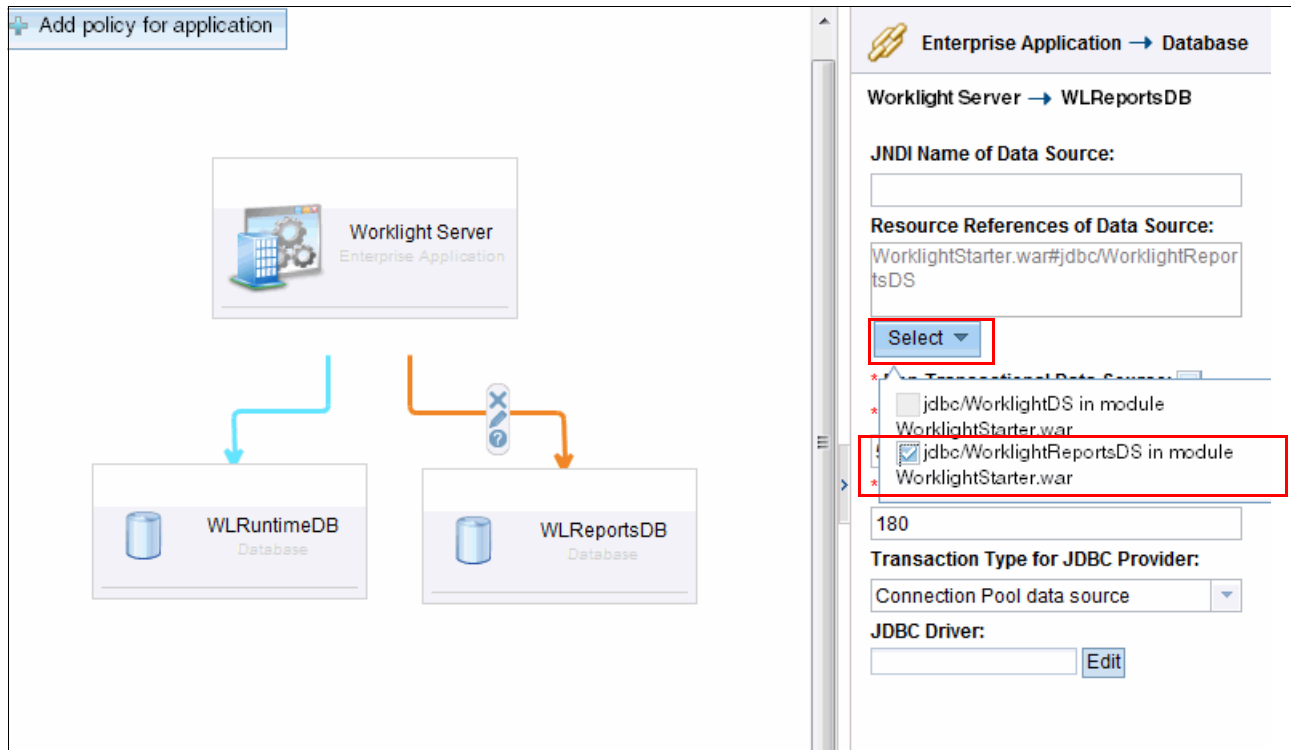


Figure 2-16 Configuring connectivity between the Worklight server and the databases

### Configuring the Worklight server to include Worklight configuration component

Complete the following steps to add the Worklight configuration component and create a link from the Worklight server component to the Worklight configuration component:

1. From the Assets list, expand Worklight Components, and then drag a Worklight Configuration component onto the canvas.
2. Create a link from the Worklight Server component to the Worklight configuration component.
3. Configure the links by using the properties that are shown in Table 2-4.

Table 2-4 Worklight configuration component properties

Property	Description
Name	Name for the Worklight configuration component (for example, WorklightConfiguration).
Worklight Console Protection	Select this option to enable security protection for the Worklight console. Clear the option to disable security protection.
Worklight Console Username	User name for Worklight console protection.
Worklight Console Password	Password for Worklight console protection

Figure 2-17 shows the window that is used for configuring the Worklight server.

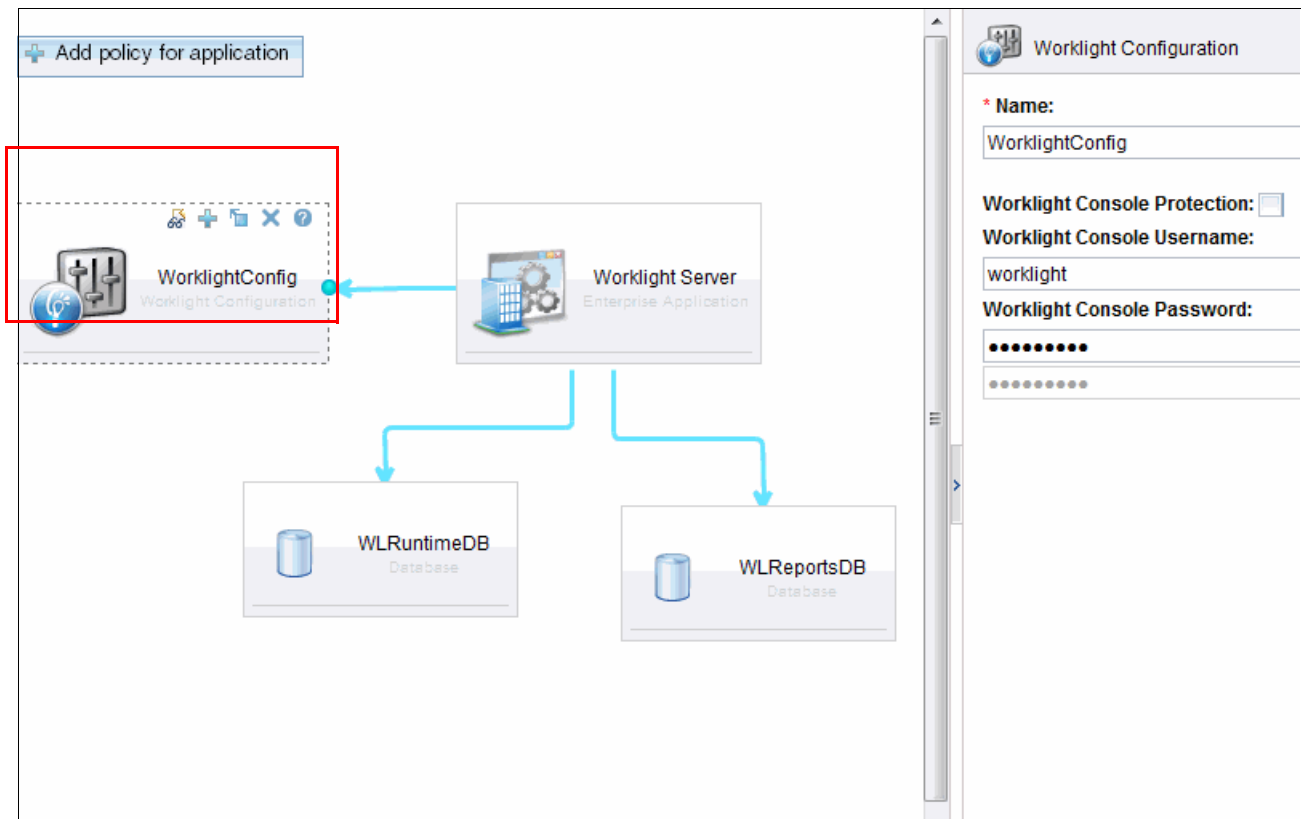


Figure 2-17 Worklight Configuration window

### Creating Worklight application and adapter

Complete the following steps to add a Worklight adapter component and a Worklight application component from Worklight:

1. From the Assets list, expand Worklight Components, and then drag a Worklight adapter component and a Worklight application component onto the canvas.
2. Configure them based on the properties that are shown in Table 2-5.

Table 2-5 Worklight adapter component and application component properties

Component	Property	Description
Worklight application	Name	Name of the Worklight application.
	Worklight application files	Worklight application files to upload. Supported formats are *.wlapp and *.zip.
Worklight adapter	Name	Name of the Worklight adapter component
	Worklight adapter files	Worklight adapter files to upload. Supported formats are *.wlapp and *.zip.

Figure 2-18 shows how to create Worklight application and adapter.

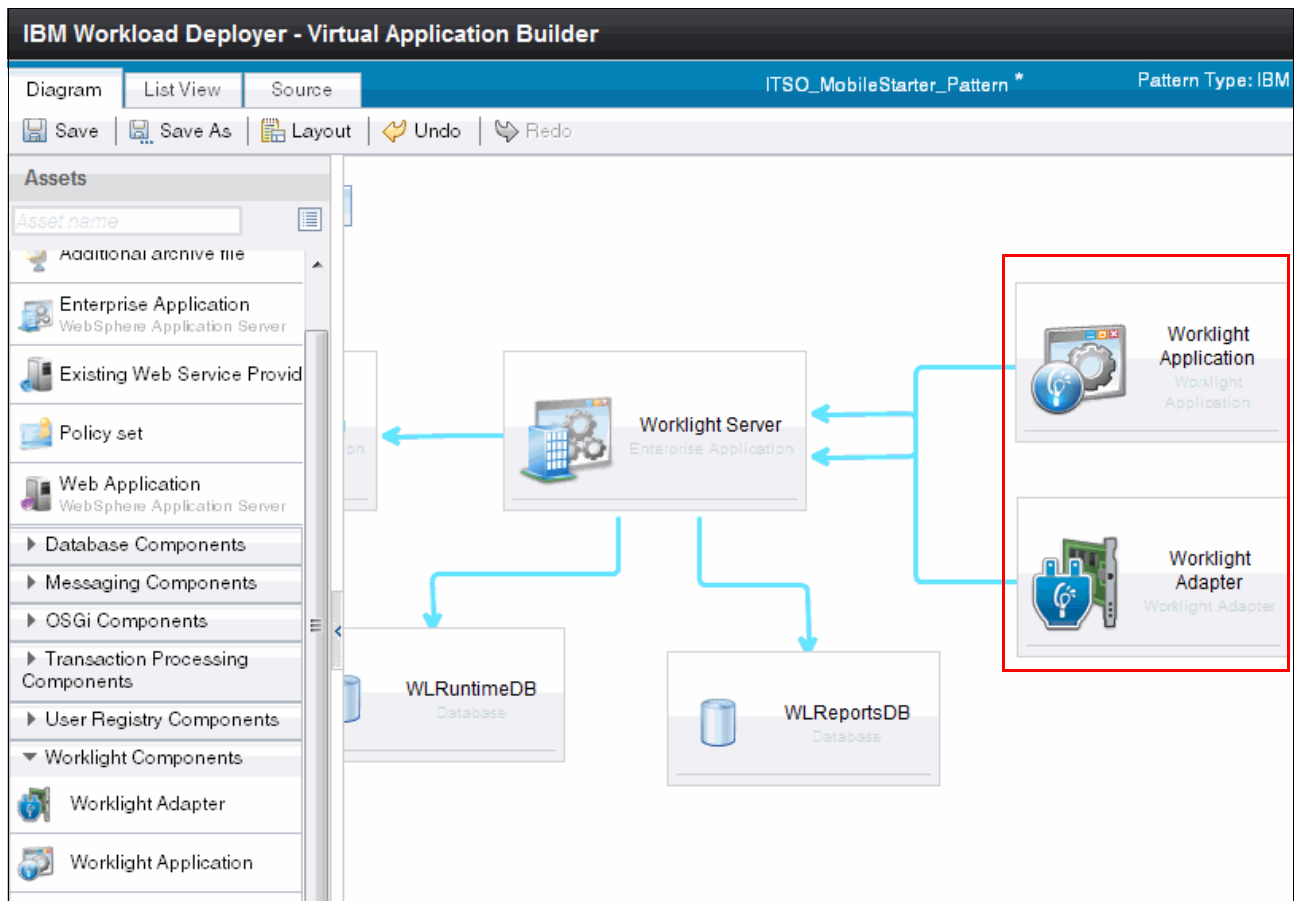


Figure 2-18 VAB window for creating Worklight application and adapter components

### Adding QoS and other policies

If you want a Worklight application to be highly available, you can add a scaling policy to the enterprise application component and specify the triggers, such as CPU-based processor usage threshold, that starts scaling your application.

When you apply a policy to an application, the policy is applied to all components in the virtual application pattern that supports the application. If you apply a policy to a specific component and apply it to the virtual application pattern as a whole, the configuration of the component-specific policy overrides the application-level policy.

This section uses the CPU-based triggers application scenarios for scaling policy in this example. Figure 2-19 on page 48 shows the VAB window for adding a scaling policy to a Worklight server.

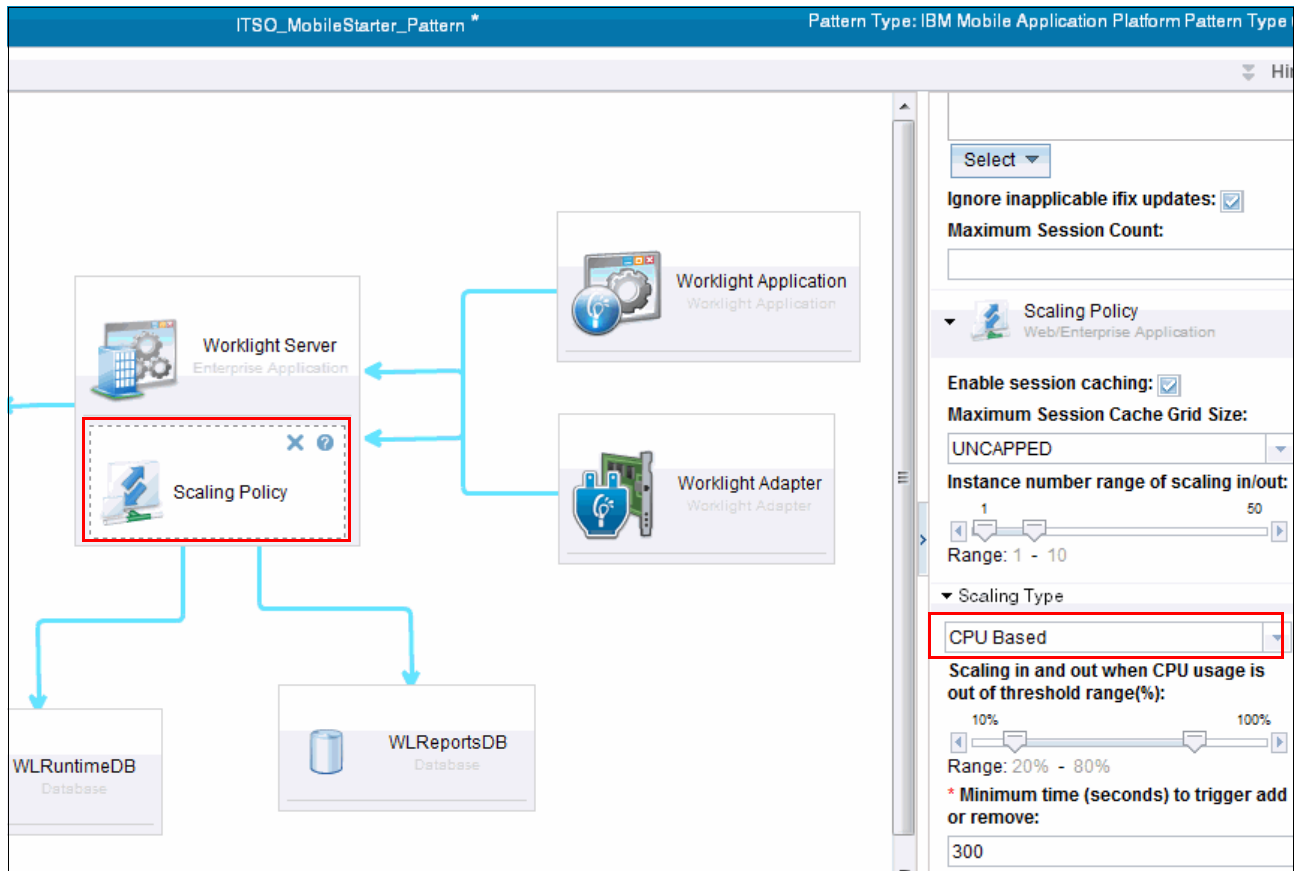


Figure 2-19 VAB window for adding scaling policies to the enterprise application component

## Creating and importing pattern plug-ins

If you need a component that is not available in the Mobile Application Platform Pattern, you can build a custom plug-in to define new components and how they behave in your environment. After the plug-in is created, you can import the plug-in into your system.

This section does not provide information about how to create a plug-in. For more information about creating a plug-in, see *Creating Composite Application Pattern Models for IBM PureApplication System*, SG24-8146, which is available at this website:

<http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248146.html?Open>

You can also see *Creating plug-ins for virtual application patterns, Part 1: An introduction*, which is available from IBM developerWorks at this website:

[http://www.ibm.com/developerworks/websphere/library/techarticles/1212\\_dejesus2/1212\\_dejesus2.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1212_dejesus2/1212_dejesus2.html)

You also can see the plug-in development guide that is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/pg\\_r\\_plugindvcomp.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/pg_r_plugindvcomp.html)

## 2.4 Design considerations

This section describes some key points for consideration when you are designing your virtual patterns. An important consideration is to determine which type of pattern is appropriate for your application and the potential reuse of the pattern.

### 2.4.1 VSP versus VAP

If you are considering which pattern to use in your environment, you must decide which pattern works best that is based on the benefits and limitations of each approach.

Table 2-6 lists the general benefits and limitations of each kind of pattern.

Table 2-6

Pattern	Benefits	Limitations
Virtual system pattern	<ul style="list-style-type: none"><li>► Provides repeatable, reproducible system deployments</li><li>► Simple to create and deploy</li><li>► More customizable than a VAP</li><li>► Greater control and administration, compared to a VAP</li></ul>	<ul style="list-style-type: none"><li>► Configuration scripts are required for customizations</li></ul>
Virtual application pattern	<ul style="list-style-type: none"><li>► Simple to create, configure, deploy, and monitor</li><li>► Easy-to-use interface</li><li>► Significant time and resource savings compared to a VSP</li><li>► Scalability functions</li><li>► Built-in shared services</li></ul>	<ul style="list-style-type: none"><li>► Less control over your environment's configuration than with a VSP</li><li>► Only web-based applications are supported</li><li>► Plug-ins are required for customizations</li></ul>

If you want to deploy your application by using a VAP, you must determine whether the application is suitable for one of the virtual application patterns that are available. Each virtual application pattern has specific compatibility criteria against which you can assess your needs.

For example, your analysis regarding whether an application can be deployed by using a particular VAP can be influenced by the programming model that is used by the application, the application type, the specifications that are used by the application, or how the application state is handled.

IBM provides assessment criteria to help you determine whether an application is compatible with a particular virtual application pattern. These criteria are presented during the onboarding process of a number of ISV applications. For more information, see the IBM developerWorks article that is available at this website:

[http://www.ibm.com/developerworks/websphere/library/techarticles/1204\\_brown/1204\\_brown.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1204_brown/1204_brown.html)

## 2.4.2 Pattern reuse

Whenever you are designing a pattern, reuse of the pattern is an important factor to consider.

Theoretically, you can create a pattern when one is needed. However, doing so leads to many nearly identical patterns, which can be challenging to manage.

The following important design principals relate to the reuse of patterns:

- Separate the application-specific behaviors from the topology.

If you design a VSP, it can be used for many types of applications that have the same topology.

For each application that has a different configuration that is related to that specific application, such as JVM configuration, application artifacts to be installed in the topology, and database schema, you must separate those kinds of specific information.

The authors recommend that virtual system patterns not include purely application-centric information in the scripts that they run. Instead, application-specific information should be read by the scripts that run in the deployment process or from instance-specific locations that are specified as part of the deployment process. You also can use an external deployment automation tool to deploy these application-specific configuration items into a pattern instance when the tool's actions are triggered by an external event, such as the completion of a new build.

This approach makes it possible to reuse each virtual system pattern you create and reduces the number of virtual system patterns that your organization must manage.

- Do not put everything in a single pattern.

Another concern is how large your pattern should be. Should all applications be contained within a single pattern, or should an application spread across multiple patterns?

The answer depends on your application and setup. In some situations, a single application might comprise only a single EAR file that is running on an application server instance. In other situations, a single application can be spread over multiple run times, including database run times and application server run times.

The authors recommend that a single pattern should contain only components that are directly related to each other and should not contain components that are shared by other components outside the pattern, unless all of those components are shared.

For example, consider a case in which you have two applications, one for internet banking and another for customer management. They each have EAR files for each application server, and both applications share common information in a shared database that is called the Customer Accounts database. However, the customer management application also relies on information from a database that is used only by the customer management application. In this case, it is best to divide the applications into the following patterns:

- Internet banking application servers and web servers
- Customer management application servers, web servers, and database
- Customer Accounts database

By using this approach, any changes that are made to the individual applications are specific to the patterns that are involved. If a change must be made to one application in a pattern, the other application continues to function without any interruption, and vice versa.



For more information about the leading practices for effectively adopting patterns, see the IBM developerWorks article *Best practices for patterns adoption in IBM PureApplication System*, which is available at this website:

[http://www.ibm.com/developerworks/websphere/techjournal/1307\\_brown1/1307\\_brown1.html](http://www.ibm.com/developerworks/websphere/techjournal/1307_brown1/1307_brown1.html)

For more information about designing a virtual system pattern, see the developerWorks article that is available at this website:

<http://www.ibm.com/developerworks/cloud/library/cl-puresystem-vspdesign/>

For more information about how the demands of PureApplication System fit into the existing roles within an IT organization, see the IBM developerWorks article that is available at this website:

[http://www.ibm.com/developerworks/cloud/library/cl-ps-aim1305\\_alignorg/](http://www.ibm.com/developerworks/cloud/library/cl-ps-aim1305_alignorg/)





# Pattern deployment

In Chapter 2, “Pattern design” on page 17, we showed how to design a pattern. Patterns provide a topology definition for repeatable deployment that can be shared.

In this chapter, we describe how to deploy a pattern on different IBM SmartCloud environments.

This chapter includes the following topics:

- ▶ Pattern deployment process
- ▶ Sources of Patterns
- ▶ Deployment on IBM SmartCloud Platforms
- ▶ Deployment solutions made easy with patterns

## 3.1 Pattern deployment process

From initial pattern design to successful deployment of a pattern, it is helpful to know the complete pattern deployment process flow, in case any debugging is needed. On every cloud system, the following four basic components are used:

- ▶ Catalog
- ▶ Pattern
- ▶ Cloud resources
- ▶ Instances

These components are shown in Figure 3-1.

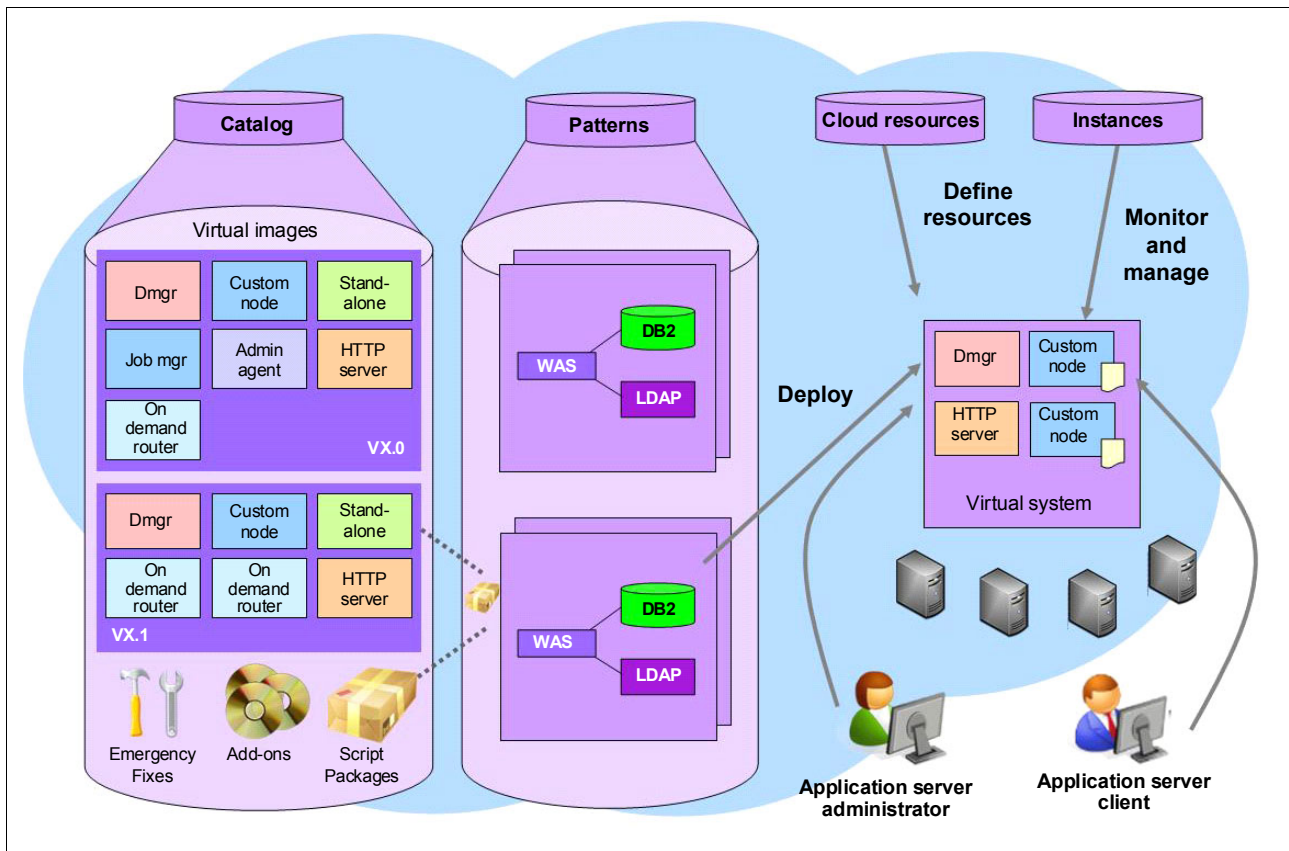


Figure 3-1 Pattern deployment model

The catalog stores content that with which you can create deployment topologies. Catalog content includes the following items:

- ▶ Virtual images
- ▶ Pattern, Pattern type
- ▶ Plug-in
- ▶ Emergency fixes
- ▶ Script packages
- ▶ Templates
- ▶ Add-ons

The content from the catalog is used as building blocks to create the patterns that describe the deployment topologies that you use in your cloud environment.

Virtual application patterns are application-centric and made up of plug-ins and patterns types. When you are designing your patterns, use applications, such as enterprise archive (EAR), web archive (WAR), and data definition language (DDL) to describe the characteristics for how the application is to be run and managed.

Virtual Application Patterns are portable across the following underlying cloud resources and cloud platforms:

- ▶ IBM Workload Deployer
- ▶ IBM SmartCloud Orchestrator
- ▶ IBM SmartCloud Application Services (SCAS)
- ▶ IBM PureApplication System

A virtual system pattern (VSP) is based on a virtual image and is made up of a group of virtual image parts. The parts that comprise the pattern are defined in the virtual image; therefore, depending on the type of virtual image that is used, different image parts are available for a pattern. For example, typical image parts for WebSphere Application Server are a deployment manager, custom node, or stand-alone application server. After virtual image parts are added to a pattern, the parts can be customized by using script packages.

Virtual appliance patterns include several preinstalled patterns that contain leading industry practices, such as clustering. You can use these patterns as-is, customize them to suit your environment needs, or build your own patterns from scratch.

VSPs are portable between cloud platforms if dependent virtual images, hypervisor editions, and Open Virtualization Archive (OVA) file are available, including sufficient cloud resources with CPU, storage, network, and memory. However, the pattern is not portable between different hypervisor technologies, such as VMware ESX, Kernel-based virtual machine (KVM), PowerVM, z/VM, and Hyper-V.

For example, if you have a VSP that uses two VMWare ESX server images, the pattern can be deployed only on an environment in which you have the dependent OVAs and supporting hypervisors.

To overcome these types of problems, open standards, such as OASIS Topology Orchestration for Cloud Applications (TOSCA) and OASIS Open Services for Lifecycle Collaboration (OSLC), can help. Cloud technologies are evolving every day and most of the problems that were encountered earlier were addressed.

Currently, VSP patterns are portable between IBM SmartCloud platform offerings, such as IBM Workload Deployer, SmartCloud Orchestrator, IBM PureApplication System and SCAS. In some cases, to avoid missing dependent OVAs, pattern providers are shipping dependent OVAs with the patterns.

As with hypervisors, cloud resources, such as network resources, IP addresses, and storage, must be defined within the system or appliance configuration. The following cloud components must be defined:

- ▶ Cloud groups
- ▶ Hypervisors
- ▶ IP groups

A cloud group is a collection of hypervisors. The cloud group must be associated with an IP group (a pool of available IP address that can be assigned to virtual systems) before you can deploy any patterns to it. Depending on the access that is defined in your Environment Profiles, you choose a cloud group as a target for the deployment when a pattern is deployed. The appliance or system then automatically places the pattern on appropriate environment within that cloud group and assigns IP addresses from the associated IP group to the virtual systems that are created.

All of the deployed patterns create instances of the patterns, where you can monitor or manage the deployed instances.

## 3.2 Sources of Patterns

There are more than 250 patterns that are certified by IBM, which includes IBM's own solution patterns, independent software vendors (ISVs) patterns and Business Partners patterns. Many companies started developing and adopting their solution (pattern) delivery for in-house and customers.

As described in Chapter 2, "Pattern design" on page 17, you can create your own patterns. A part from the patterns that you create and the default patterns that are included with the system, you can get patterns from other resources, including ISVs, who built their entire solution stack as patterns. As part of the PureSystem ecosystem drive, IBM enabled and certified more than 200 ISVs applications for IBM PureSystems environment.

For an overview of patterns that were created by IBM or a Business Partner, see the IBM PureSystems Centre, which is available at this website:

<http://www-01.ibm.com/software/brandcatalog/puresystems/centre/browse>

The PureSystems Centre offers a simplified experience for PureSystems users to obtain PureSystems optimized content, fixes, updates, and access to IBM and IBM Business Partner expertise.

Use the PureSystems Centre site to access solutions from IBM and IBM Business Partners, updates to systems and solutions, and expertise for maximizing the benefit of systems and solutions.

## 3.3 Deployment on IBM SmartCloud Platforms

When you are working in different IBM SmartCloud environments, such as IBM Workload Deployer, IBM PureApplication System, SCAS and SmartCloud Orchestrator, IBM unified the user experience in the tasks and user interfaces. However, depending on the capabilities of the individual products, some of the windows and user interfaces can differ. We can access each of the IBM SmartCloud environments through the following interfaces:

- Web-based user interface

The primary access for the SmartCloud platforms is through the web-based user interface, which provides administrator and users with a wizard-based setup environment. The web-based UI includes common command features and tabs across all of the SmartCloud platforms, apart from some extra capability tabs on IBM PureApplication System and SmartCloud Orchestrator. Therefore, a user who worked on one SmartCloud environment can adapt to other SmartCloud environment with ease.

► Command-line interface

The command-line interface (CLI) provides a scripting environment that is based on Jython, the Java based implementation of Python. In addition to commands that are specific to IBM SmartCloud platforms, you can issue Python commands at the command prompt. You can use the CLI to manage an IBM SmartCloud platform remotely. The CLI communicates with the IBM SmartCloud platforms over an HTTPS session. CLI can run in interactive and batch modes.

The CLI tool can be downloaded from the IBM SmartCloud platform's web-based user interface to a Windows or Linux system.

**Note:** On each platform, the names of the tool and command differs, but they perform the same job. Internally, it uses the Representational State Transfer (REST) API to communicate.

► REST API

The IBM SmartCloud platforms provide a subset of its functions by using a REST API. The REST API also provides a means to interact with the platform that is language-neutral and programming model-neutral. When the REST API is used, you interact with the resources of the platforms, such as the hypervisors, patterns, and script packages, by using well-defined HTTP URLs and associated HTTP verbs (GET, POST, PUT, and DELETE). Unlike the web-based UI, the REST API is supported over the HTTPS protocol only. The appliance uses a self-signed certificate for its SSL sessions. The same certificate is used for the web-based UI, CLI, and REST API sessions. You must configure your HTTPS client to accept or ignore this certificate during the SSL handshake. You must use an HTTPS client that you can use to set the HTTP headers for each request.

The REST API also supports the sending and receiving of UTF-8 encoded data only. Ensure that your HTTP client is appropriately set to encode and decode character data, including JSON data.

### 3.3.1 IBM Workload Deployer

IBM Workload Deployer is one of the foundational elements of the IBM private cloud strategy. This appliance provides rapid adoption and deployment of Infrastructure as a Service (IaaS) and Platform as Service (PaaS) offerings. It provides a proven path to use existing IT resources by improving the efficiency and flexibility of these infrastructures. IBM Workload Deployer is a solution for organizations that seek agility in response to a dynamic business environment.

#### IBM Workload Deployer deployment environment

The cloud environment that patterns are deployed to consists of a set of hypervisors that you provide. You configure the hypervisors to the IBM Workload Deployer, then create pools of IP addresses to assign to provisioned systems on those hypervisors. Each hypervisor belongs to a cloud group. When a virtual system pattern (VSP) or virtual application pattern (VAP) is deployed, it is deployed to a cloud group. The IBM Workload Deployer selects one or more hypervisors in the cloud group and assigns IP addresses to the provisioned systems from the IP group for that hypervisor. An alternative to having a static set of IP addresses is to use an environment profile to specify the IP address that is to be assigned by the pattern deployer, as shown in Figure 3-2 on page 58.

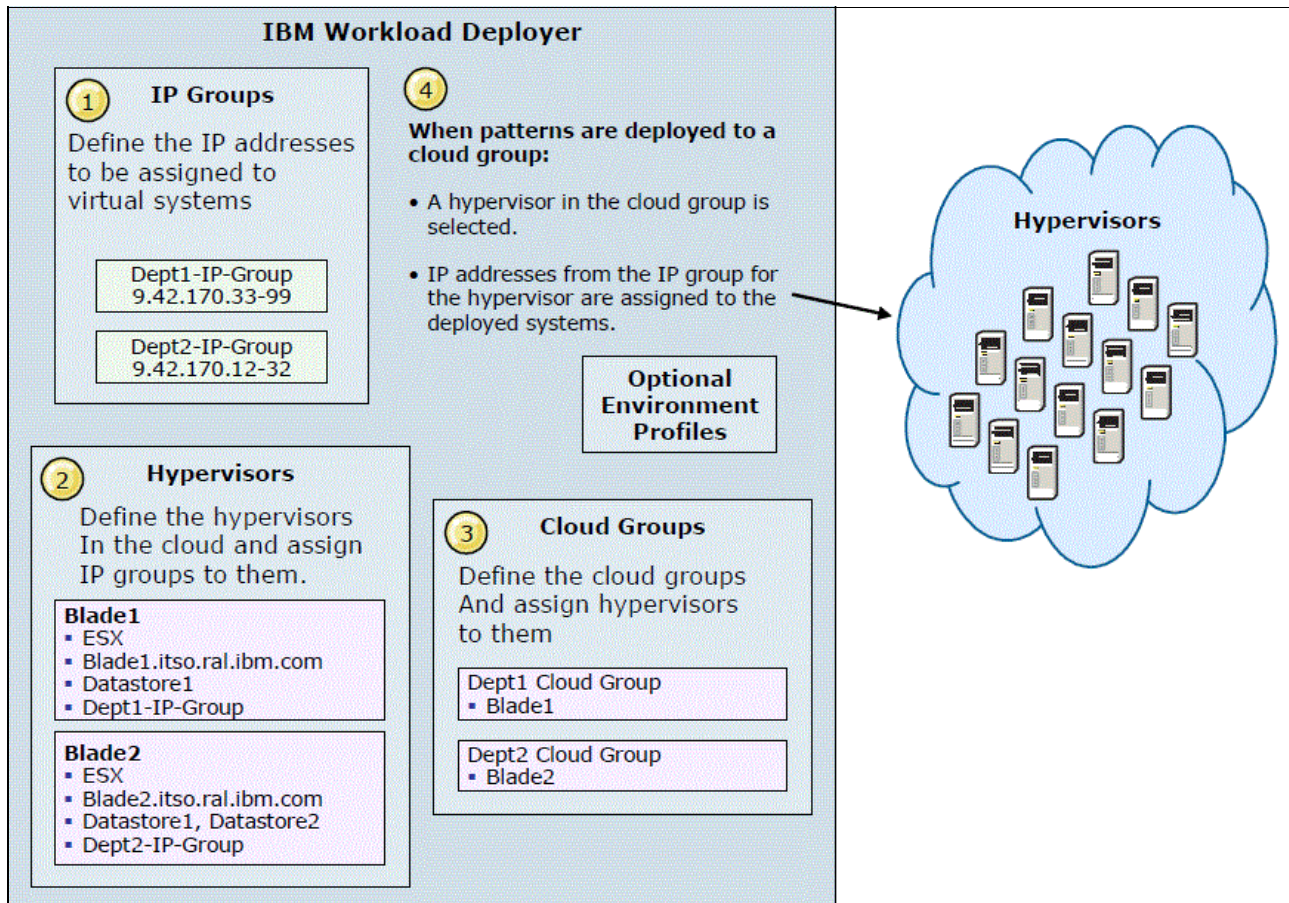


Figure 3-2 Cloud configuration

### Hypervisors

A hypervisor is a software virtualization program that provides a layer of abstraction between operating systems and physical resources on a machine. This abstraction layer allows multiple operating systems and application stacks to run on a single physical entity and share resources, which provides higher levels of resource usage.

To set up the cloud, the administrator defines the location and login credentials for the hypervisors. These hypervisors host the VSPs and VAPs that the IBM Workload Deployer dispenses. IBM Workload Deployer automatically detects the storage that is associated with the hypervisors and manages the placing of the middleware virtual systems across the set of hypervisors. The following hypervisors are supported:

- ▶ VMware ESX
- ▶ IBM PowerVM
- ▶ IBM z/VM

**Note:** For the examples in this book, we use the VMware ESX environment.

### IP groups

Pools of IP addresses, which are known as *IP groups*, are configured for use by the deployed virtual machines. The administrator defines this pool of IP addresses, and when virtual machines are created, the appliance assigns each machine a unique value. IP addresses then can be added to and removed from the pre-configured pool, as needed.



**Tip:** Before your pool of IPs is defined, ensure that you defined the IBM Workload Deployer client to the DNS server, that your defining IP addresses are free, and that all of the IPs include name resolution (DNS).

### ***Cloud groups***

A cloud group is a collection of related hypervisors. When patterns are deployed to create virtual systems, you use a cloud group as the deployment target. One or more hypervisors of the same type make up a cloud group, for example, you can group all of your ESX hypervisors together or all of your high-end PowerVM hypervisors together. You can manage resource allocation thresholds, such as processor, memory usage, and also verify the runtime status of your configured hypervisors.

### ***Environment profiles***

Environment profiles relate deployment configurations, such as virtual machine names, IP address assignment, and cloud groups. Deploying patterns that use environment profiles allows deployments across environments by using a single pattern. In IBM Workload Deployer, environment profiles provide the following functionality:

- ▶ Define the operational environments, such as development, test, or quality assurance
- ▶ Define virtual machine naming conventions within the operational environment
- ▶ Specify whether the IP group or a pattern deployer provides the IP address on the deployment
- ▶ Segment the clouds and IP groups within the clouds to specific environments
- ▶ Assign aliases to the cloud resources, such as clouds and IP groups
- ▶ Assign sections within the clouds to specific users or groups
- ▶ Define limitations on the number of virtual processors, virtual memory, and storage

With environment profiles, you can also group multiple clouds to be used in the deployment. You can deploy a pattern to multiple cloud groups of the same hypervisor type. For example, you might deploy a pattern to multiple PowerVM cloud groups. However, you cannot deploy a single pattern to a z/VM cloud group and to a PowerVM cloud group. Environment profiles are platform-specific, so IBM Workload Deployer filters out the appropriate clouds.

### ***Considerations with IBM Workload Deployer***

This chapter does not cover installation and setup details. It is assumed in this book that all of the required environments are in place and well-configured.

As an administrator, ensure that the following tasks are complete:

- ▶ Configure IP groups, hypervisors, and cloud groups.
- ▶ Environment Profiles are optional but help to manage resources and access rights (suggested task).
- ▶ Accept all of the licenses for entitled images and patterns.
- ▶ Make sure for the pool of IP addresses all addressees are free for IBM Workload Deployer to use and have Domain Name (DNS) Resolution.
- ▶ Configure the IBM Workload Deployer and Hypervisors as a client to the NTP and DNS servers.
- ▶ Create users and groups with proper access credentials.
- ▶ If you want to upload, export and import patterns (VSP, VAP, and OVAs) and their size is more than 2 GB, use the CLI tool only.

- ▶ You can export OVAs only to an SSH server over the GUI.
- ▶ You can import pattern and OVAs to IBM Workload Deployer over GUI from only an HTTP server. FTP, SFTP, and NFS are not currently supported.

As a user, consider the following points before you start:

- ▶ Make sure that you have the required access credentials.
- ▶ If you want to upload, export, or import patterns (VSP, VAP and OVAs) and the size is greater than 2 GB, use CLI tool only.
- ▶ You can export OVAs over GUI to only an SSH server.
- ▶ You can import patterns and OVAs to IBM Workload Deployer over GUI from an HTTP server and only an HTTP Server. FTP, SFTP, and NFS are currently not supported.

### ***Deploying patterns by using the web-based interface***

The primary access to the IBM Workload Deployer appliance is through the web-based user interface. This management console is enabled when the appliance is first started, as shown in Figure 3-3.

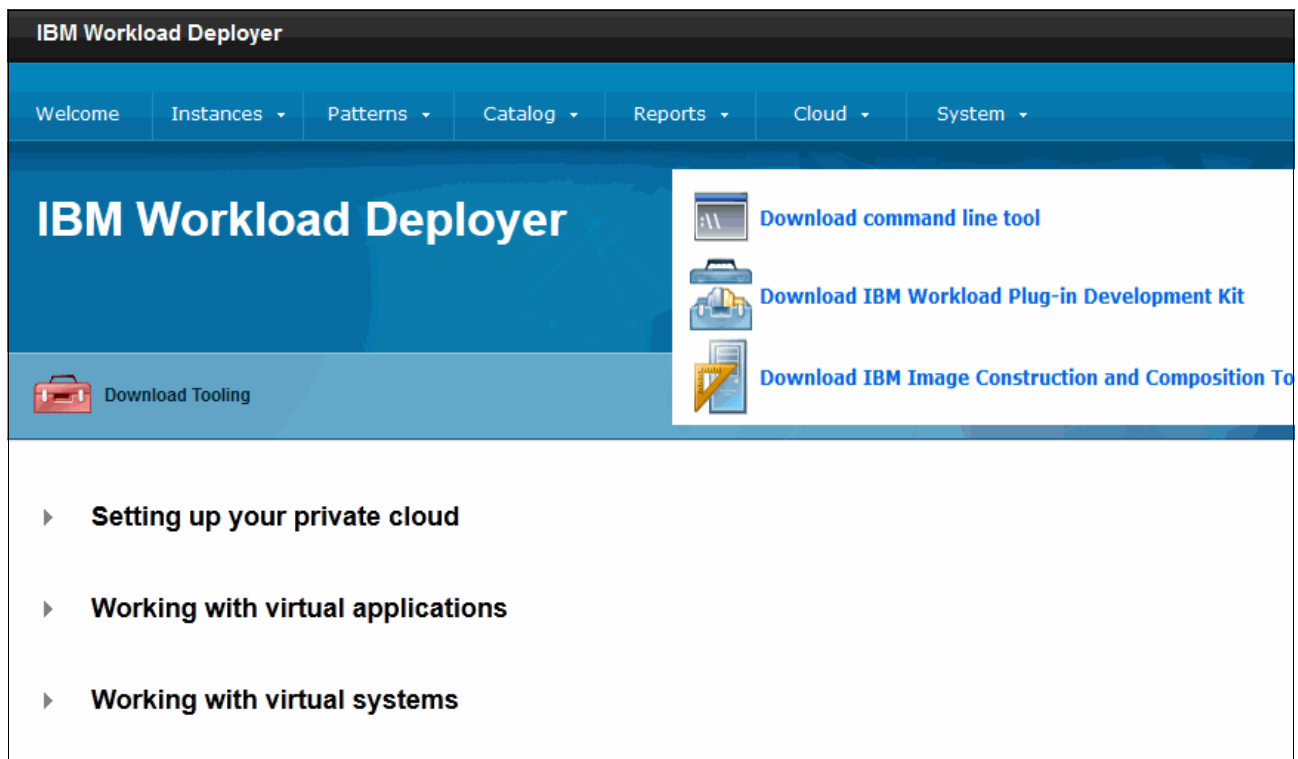


Figure 3-3 IBM Workload Deployer web interface

The Welcome window provides wizards with which you can configure the core functionality of IBM Workload Deployer in a step-by-step approach. There are also drop-down menus that can be used in a more granular way. The menu items are grouped by category. For example, the IBM Workload Deployer settings, user, group management, system security, monitoring, and troubleshooting are under the System menu, and configuration options, such as hypervisors, cloud, and IP groups, are under the Cloud menu.

For example, to deploy a pattern, click **Pattern** → **Virtual Systems**, which gives you a list of patterns on left side of the window. Select the pattern that you want to deploy, as shown Figure 3-4 on page 61.

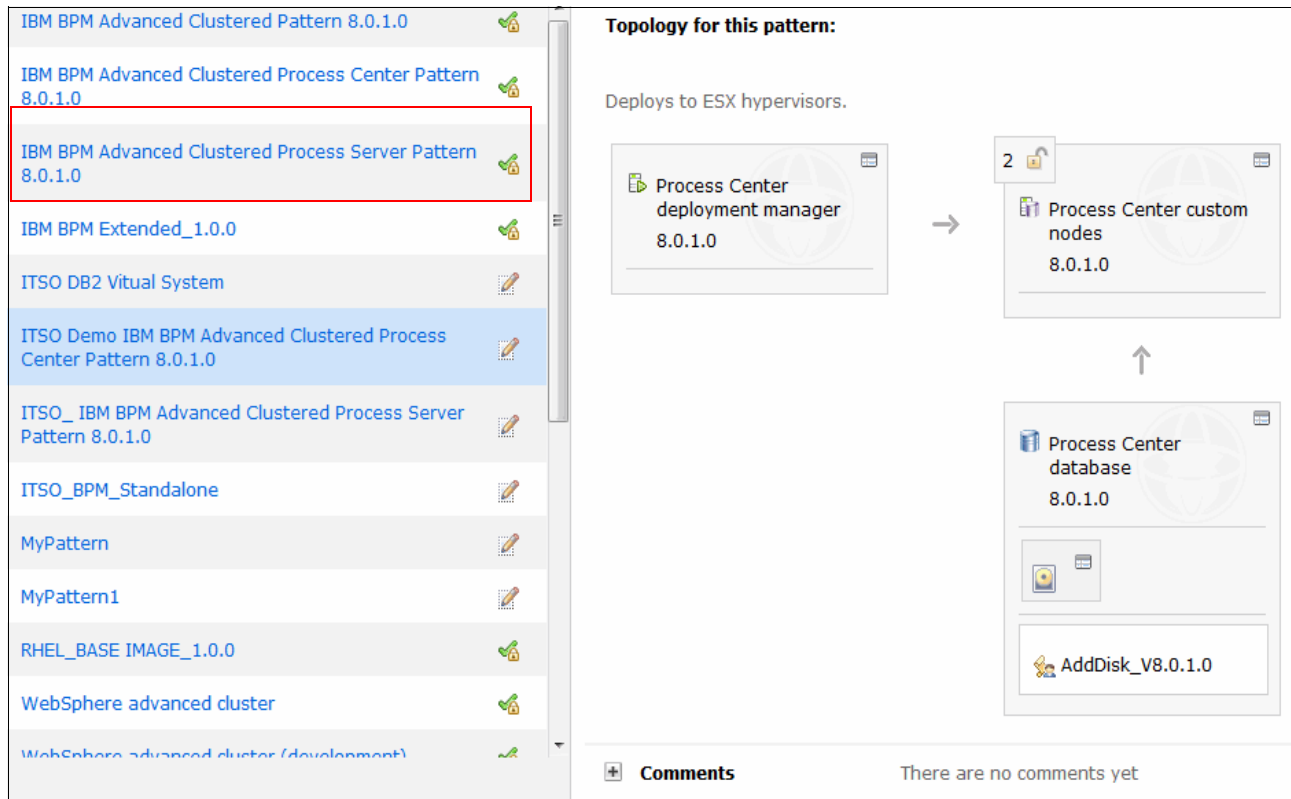


Figure 3-4 VSP patterns from web interface

This example used the Virtual System Pattern, IBM BPM Pattern. To modify the pattern, click **Edit** in the upper right of the canvas. After you are ready to deploy the pattern, click **Deploy** in the upper right of the canvas. When you click **Deploy**, another window opens in which you can select the deployment environment, as shown in Figure 3-5. You can name the deploying pattern, choose an environment depending on your access rights, schedule the deployment, and configure the individual part's deployment variables.

The dialog box is titled 'Describe the virtual system you want to deploy.' and contains the following elements:

- Virtual system name:** A text input field containing 'ITSO BMP demo'.
- Choose environment:** A link with a checkmark icon.
- Schedule deployment:** A link with a checkmark icon.
- Configure virtual parts:** A link with a checkmark icon, followed by a list of components, each with a checkmark icon:
  - Process Center custom nodes
  - Process Center database
  - IBM HTTP Server for Process Center
  - Process Center deployment manager

At the bottom are 'OK' and 'Cancel' buttons.

Figure 3-5 IBM BPM VSP Deployment

After selecting **OK**, the system deploys the pattern. The amount of time that is needed depends on the network speed, hypervisor system configuration, components that are used, and the script sizes in the pattern. You can track the status of the deployment by clicking **Instances** → **Virtual Systems**. After the pattern is deployed successfully, you see a green arrow mark, as shown in Figure 3-6.

IBM Workload Deployer

Welcome

Instances

Patterns

Catalog

Reports

Cloud

System

Virtual System Instances

Search...

ITSO Demo BPM

ITSO Demo BPM

Created on:

Aug 6, 2013 9:55:27 PM

From pattern:

ITSO Demo IBM BPM Advanced Clustered Proce

Using Environment profile:

None provided

Updated on:

Aug 7, 2013 12:41:09 AM

Current status:

The virtual system has been deployed

Access granted to:

cbadmin [owner]

Add more...

Snapshot:

(none)

Create

History

The virtual system has been deployed

Figure 3-6 IBM BPM VSP Deployment status

If you want to know the number of VMs that are deployed and their individual status, you can browse in the same page, as shown in Figure 3-7.

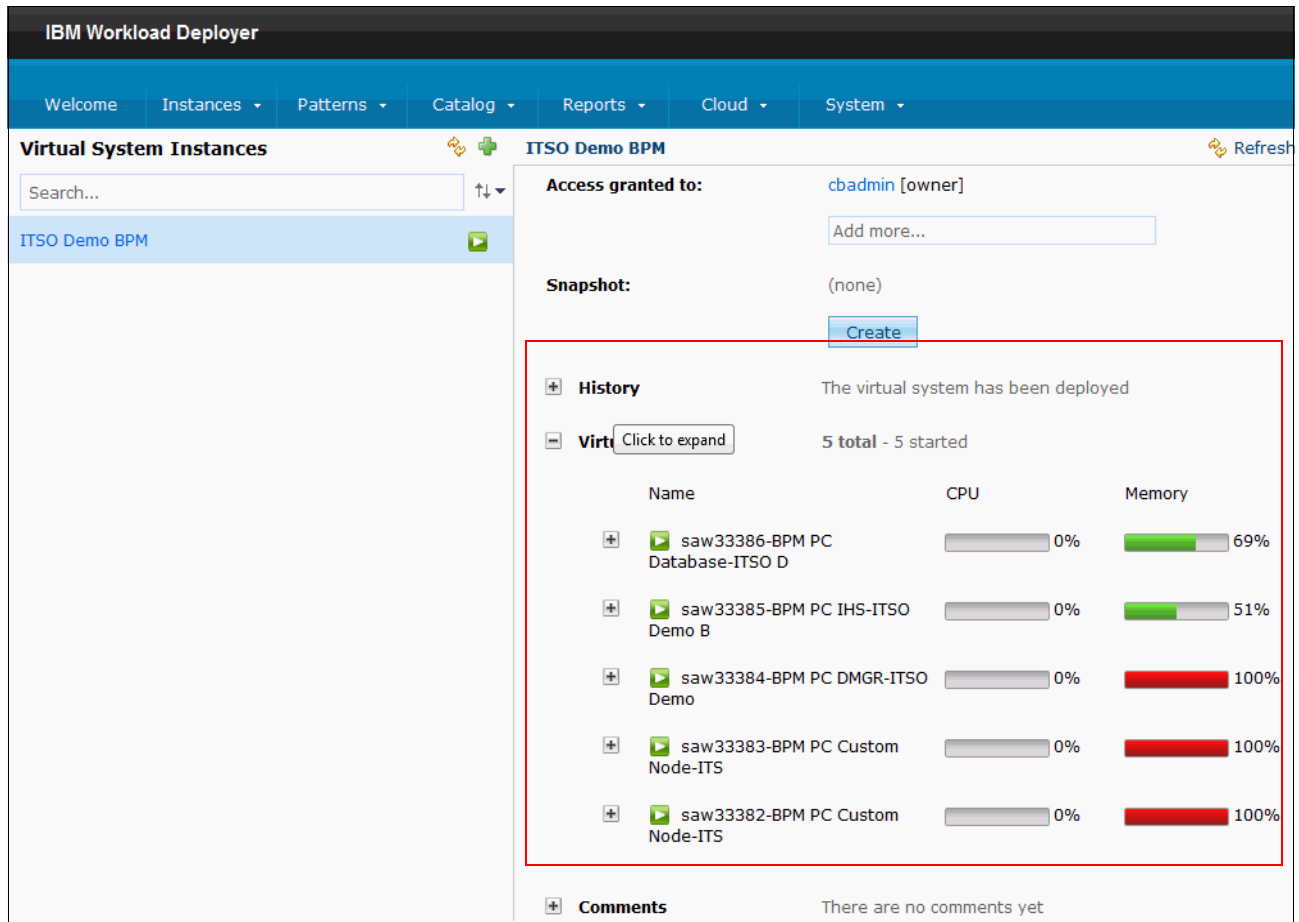


Figure 3-7 IBM BPM VSP Deployment: VMs status

You can stop, shut down, and start the deployed instance by selecting the appropriate button in the upper right corner of the user interface. You can also perform system and application service fix pack management tasks by clicking **Service** in the upper right corner of the window.

After you know the status of the instance and VMs, you can access the services that are deployed. There are links to each service console that you defined in the pattern. You can access a console by expanding the individual servers. In our example, the console we are accessing is saw33383-BPM PC Custom Node-ITS, as shown in Figure 3-8 on page 64.

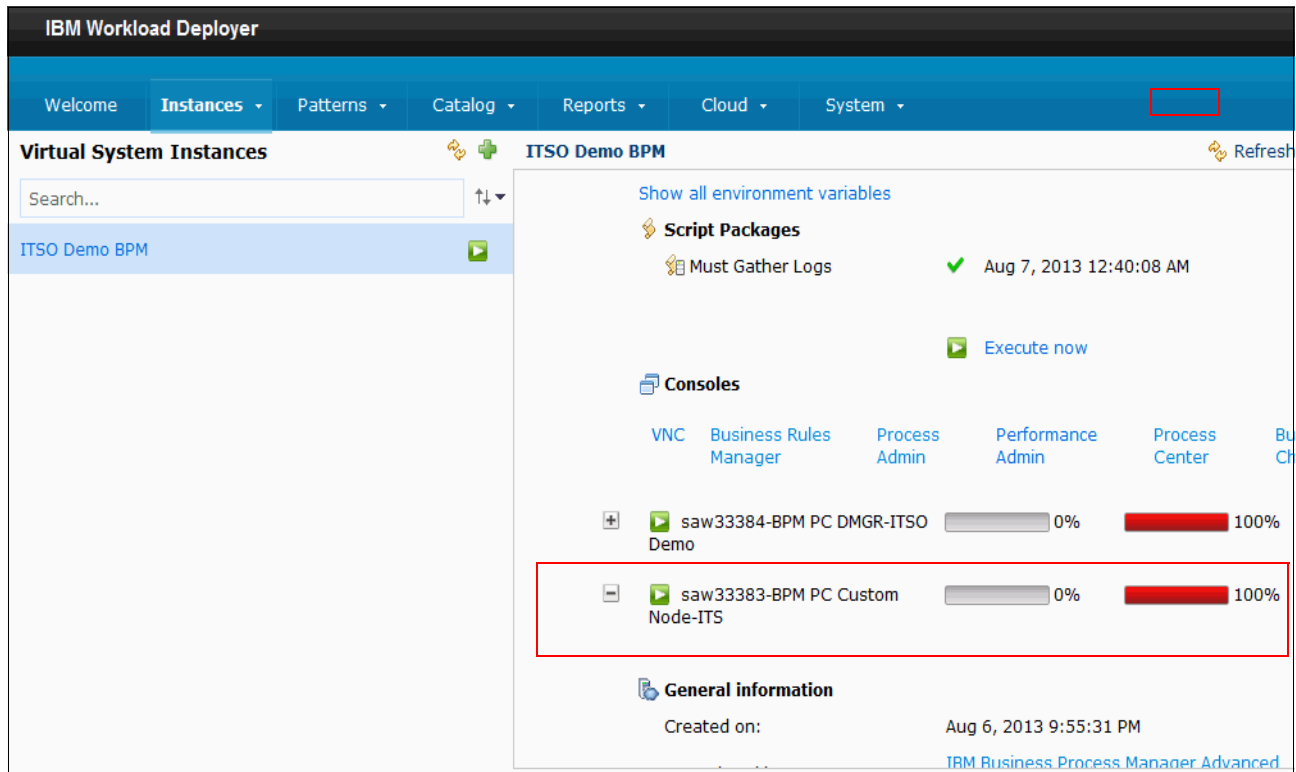


Figure 3-8 IBM BPM VSP Deployment: Service Consoles

Click any of the service links to start using them.

### Deploying patterns by using the CLI

Similar to the web-based interface, you can deploy a VSP pattern by using the CLI. On the CLI, you can deploy patterns interactively by using sample scripts that are provided with the CLI tool, as shown in Example 3-1.

#### Example 3-1 Deploying patterns by using the command prompt

```
#./deployer -h IWD.itso.ral.ibm.com --acceptcert -u iwduser1 -p userpasswd -f
../samples/deployPattern.py
It will prompt for Name for the deploying VSP
new virtual system name: ITSO Demo BPM
```

Then its going to list available pattern to deploy and wait for your input like

1. WebSphere single server
2. WebSphere cluster
3. WebSphere cluster (development)
4. WebSphere cluster (large topology)
5. WebSphere advanced cluster
6. WebSphere advanced cluster (development)
7. DB2 Enterprise 9.7
8. DB2 Enterprise 10.1
9. DB2 Enterprise 9.7 HA cluster
10. DB2 Enterprise 9.7 DR cluster - Standby (1st)
11. DB2 Enterprise 9.7 DR cluster - Primary (2nd)
12. DB2 Enterprise 10.1 HA cluster
13. DB2 Enterprise 10.1 DR cluster - Standby (1st)
14. DB2 Enterprise 10.1 DR cluster - Primary (2nd)

15. Advanced Middleware Configuration v1.0.0.3  
16. BPM Advanced Clustered Pattern 8.0.0.0  
17. BPM Advanced Clustered Process Center Pattern 8.0.0.0  
18. BPM Advanced Clustered Process Server Pattern 8.0.0.0  
19. IBM SOA Policy Pattern 2.0.0.0  
select a pattern to deploy: 17

Then it will list the Profiles available and wait for your input like

1. Test Profile1  
2. Test Profile2  
select an environment profile: 1

Then it will prompt for scheduling the start event of the pattern like

number of seconds to delay start (default=no delay): <Provide time and enter>

Then it will prompt for scheduling the stop event of the pattern like

number of seconds until stop (default=no scheduled stop): <Provide time and enter>

then it will prompt for root and virtuser password like

root password for virtual machines:  
user password for virtual machines:

once it started the instance it will provide the message like  
virtual system created:

```
{
  "acl": (nested object),
  "created": Aug 7, 2013 10:04:24 PM,
  "currentmessage": None,
  "currentmessage_text": None,
  "currentstatus": "RM01036",
  "currentstatus_text": "Queued",
  "desiredstatus": "",
  "desiredstatus_text": "",
  "environmentprofile": (nested object),
  "id": 68,
  "maintenances": (nested object),
  "name": "BPM demo",
  "owner": (nested object),
  "pattern": (nested object),
  "priority": 4,
  "snapshots": (nested object),
  "updated": Aug 7, 2013 10:04:24 PM,
  "virtualmachines": (nested object)
}
#
```

---

### 3.3.2 SmartCloud Orchestrator

IBM SmartCloud Orchestrator provides comprehensive automation of cloud services, supporting infrastructure and application and platform services. They are delivered through a self-service portal and automated by a workload editor that is integrated with the orchestration engine. Application topologies are graphically composed in the workload editor and then linked to a wide set of automation workflows that are built with a sophisticated yet intuitive orchestrator. This combination maximizes speed and flexibility to build and deliver cloud services.

#### IBM SmartCloud Orchestrator architecture

IBM SmartCloud Orchestrator is a comprehensive product that integrates the capabilities of several other IBM solutions. It adds several components and functionalities to IBM SmartCloud Provisioning.

The main components of IBM SmartCloud Orchestrator are the process engine and the corresponding modeling user interface, which is used to create processes. For this purpose, SmartCloud Orchestrator uses the capabilities of IBM Business Process Manager. It also integrates other domain-specific components that are responsible for functions, such as monitoring, metering, and accounting. SmartCloud Orchestrator bundles all of these products and components and provides the processes that are required to implement the domain-specific functionalities. The self-service user interface and the administrative user interface of SmartCloud Provisioning are extended to reflect the new functions that are provided by SmartCloud Orchestrator, as shown in Figure 3-9.

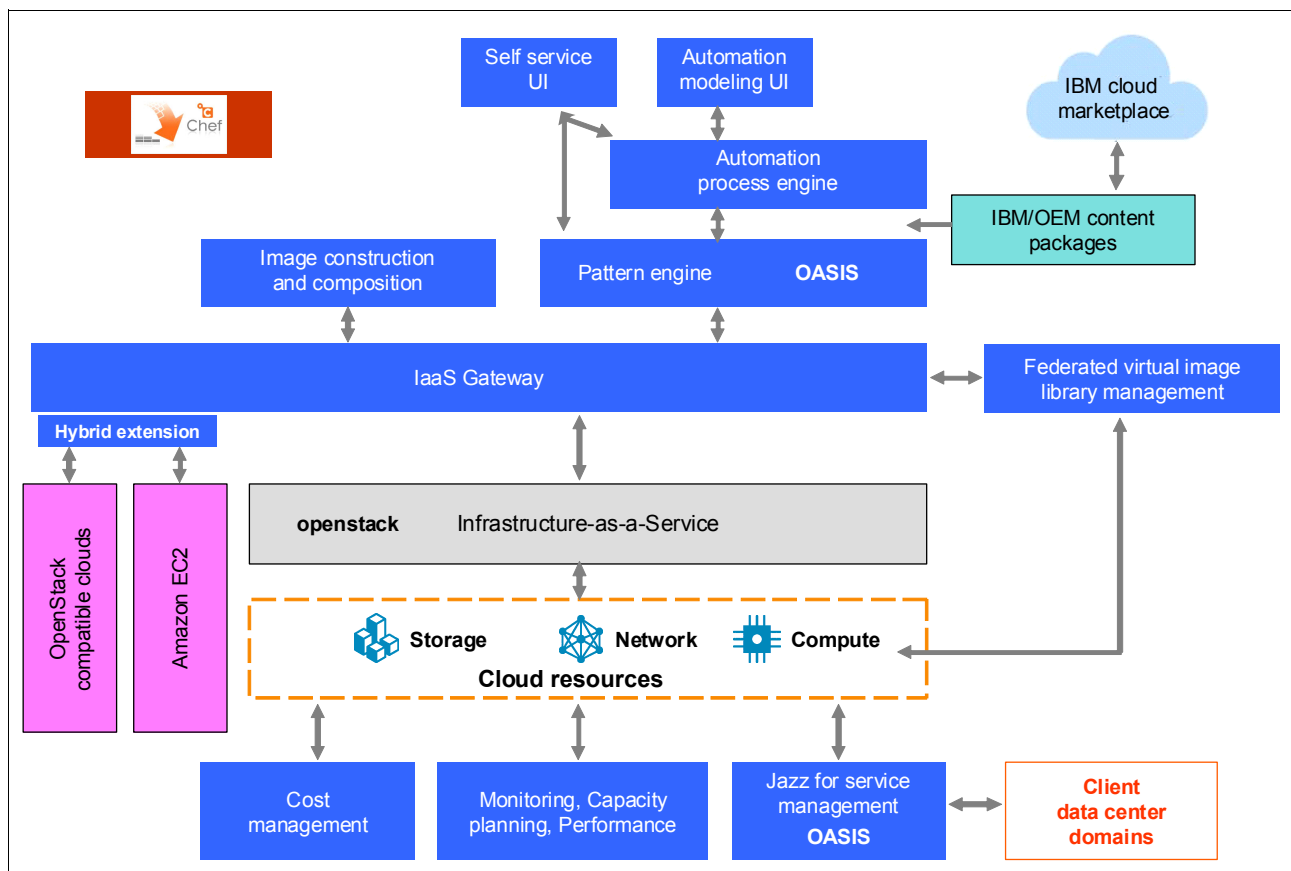


Figure 3-9 SmartCloud Orchestrator architecture



SmartCloud Orchestrator features the following major components:

► IaaS

IaaS is responsible for managing access to compute, storage, and networking resources in the virtual environment. All requests to provision services are managed by this component. The IaaS component is delivered by using OpenStack, a leading open source, community-driven project, for highly scalable and highly resilient cloud infrastructure management.

► IaaS gateway

The IaaS gateway provides a routing mechanism that enables the following configurations:

- Multi-geographical deployments in which multiple IaaS and OpenStack instances can be deployed in multiple sites and then federated together within SmartCloud Orchestrator.
- Connectivity to public clouds, such as Amazon EC2 and other public clouds that are compatible with OpenStack.

► Software stacks

Although they are not a specific component, software stacks represent the concept that when one or more virtual systems are deployed, multiple software packages can be specified to be deployed upon first boot of those systems. It can be done by starting simple installation scripts, but other strong tools can be used, such as chef recipes and cookbooks for automated installation and configuration.

► Patterns

Patterns allow for deploying more complex middleware configurations and multinode applications. The patterns component provides a graphical editor with which the user can describe multiple virtual systems, each with a base image and set of software to be installed, and then specify the relationships and configuration scripts that are necessary to connect those systems. With this level of automation, an entire multisystem deployment can be done with a few clicks.

► Image management

The Image management component features the following major functional areas:

- The Image Construction and Composition Tool (ICCT), with which a user can describe a base image and set of software stacks and then capture that fully deployed system into an image that can be reused.
- The Virtual Image Library (VIL), with which a user can manage images that are similar to other assets. This provides check-in and check-out options, versioning, indexing, and searching facilities so that corporate compliance rules can be followed, even when the images are offline.

► Workflow orchestration

This component provides a graphical editor with which the user can easily customize and extend the procedures that are followed when a user request is started. It also provides the facilities to customize the self-service catalog so that users have access to various service request types that they can access. This component is delivered by embedding IBM's award-winning Business Process Manager technology with a number of pre-built automation toolkits that make it possible to integrate workflow automation with the cloud platform and its other components. The graphical designer is highly flexible, which provides many integration techniques that range from the invocation of simple scripts and calling web services to starting more sophisticated programs, such as those written in Java.

- Cloud marketplace

The cloud Marketplace is a publicly accessible website from where various forms of automation can be downloaded and used within SmartCloud Orchestrator. It includes references to supported automation communities, such as Chef, ready to use Patterns and Images, and various pre-built Workflow Orchestration routines, packages, and toolkits. It is designed to “ship when ready”, which means that new automation can become available at any time, regardless of SmartCloud Orchestrator release schedules.

- Service management

This component is optional extra management functions that are included in SmartCloud Orchestrator Enterprise. It also highlights the ability to integrate through Workflow Orchestration other management tools and disciplines that might be important within your environment.

- Development tools

By using these tools, developer tools from IBM Rational Team Concert™ can be integrated with a set of plug-ins within SmartCloud Continuous Delivery so that a user can automate a continuous delivery pipeline from the checking in of code through build, deployment, test, and promotion. These tools are not provided within SmartCloud Orchestrator.

**Note:** On SmartCloud Orchestrator, you can have many more functionalities for applying Business process around pattern and cloud resource. However, in this book, only pattern and its usage are described.

For more information about IBM SmartCloud Orchestrator installation and configuration, see the IBM SmartCloud Orchestrator information center at this website:

[http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc\\_2.2/welcome.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc_2.2/welcome.html)

### 3.3.3 IBM PureApplication System

The PureApplication System is an integrated, highly scalable system that provides an application-centric computing model in a cloud environment.

An application-centric system is an efficient way to manage complex applications and the tasks and processes that are started by the application. The entire system implements a diverse virtual computing environment in which different resource configurations are automatically tailored to different application workloads. The application management capabilities of the PureApplication System platform make deployment of middleware and other application components quick, easy, and repeatable.

PureApplication System delivers the following components, functions, and capabilities as one integrated system:

- Virtualized system and application workloads:
  - Integrated middleware, such as IBM WebSphere Application Server, web server, DB2, and hypervisor images
  - Elastic data
  - Application-centric workloads that are created by using pattern types, such as web application patterns, database application patterns, and topology patterns.

- ▶ Scalable infrastructure:
  - Optimized hardware tuned for running workloads
  - Isolated networking for secure communications
  - Server resiliency to prevent overload or failures
  - Dynamic storage
- ▶ Integrated delivery:
  - Factory assembled and wired system
  - Tuned for maximum efficiency of data, storage, workload execution, and retrievability
  - Simple approach to managing all integrated components and monitoring health of the system
  - Single pane of glass management for administrator and application deployment

### ***IBM PureApplication System Deployment Environment***

When customers receive an IBM PureApplication System, initial system configuration is done by the IBM engineer by using a process that is called System Genesis (SGEN). This process is required to get the IBM PureApplication System operational with customer infrastructure. SGEN is a four-hour configuration and setup process. When it is completed, the system is ready for use. The SGEN Process involves defining the VLANs that are going to be used (customer and management) and the ports on the Top Of the Rack (TOR) switches that must be enabled.

The SGEN process goes smoothly with a completed Technical Design Analysis (TDA). All information that must be entered in SGEN process is based on of the TDA. After the SGEN is completed, you receive an administrative and superuser password.

### **Post SGEN Tasks**

The following steps must be completed on PureApplication System:

- ▶ Creating cloud and IP groups entries by using the IBM Engineer Account (IBMeng)
- ▶ Configuring cloud and IP groups as administrator account
- ▶ Creating an environment profile
- ▶ Configuring DNS

Figure 3-10 on page 70 show a configured PureApplication System.

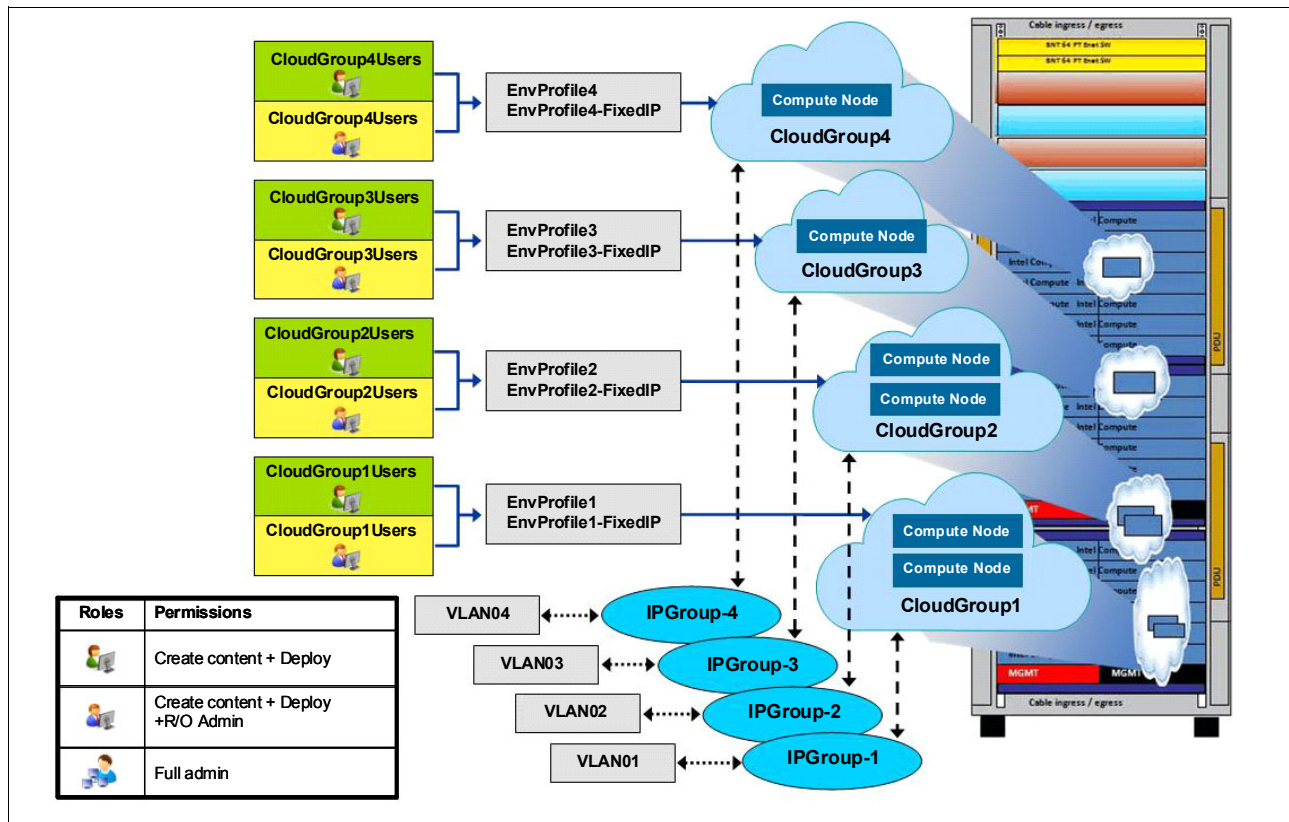


Figure 3-10 Configured PureApplication System

### Considerations for PureApplication System

The details of the setup are not described in this section. It is assumed that all of the required environments are in place and well-configured.

As an administrator, ensure that the following tasks are complete:

1. Configure IP groups, hypervisors, cloud groups.
2. Create environment profiles so you can manage the resources and access rights.
3. Accept all the licenses, entitled images, and patterns.
4. Ensure the pool of IP addresses in the IP Groups you use are all free for use by PureApplication System and have Domain Name (DNS) resolution.
5. Configure NTP and DNS Servers.
6. Create users and groups with proper access credentials.
7. If you want to upload, export, or import patterns (VSP, VAP, or OVAs) with sizes greater than 2 GB, use the CLI tool only.
8. Export OVAs over GUI to an SSH server only (optional).
9. Import pattern and OVAs over GUI from HTTP server only. FTP, SFTP, and NFS are not currently supported (optional).

As a user, ensure that the following tasks are completed before you begin:

1. Ensure that you have required access credentials.
2. If you want to upload, export, or import patterns (VSP, VAP, and OVAs) with size greater than 2 GB, use the CLI tool only.

3. Export OVAs over GUI only to an SSE server (optional).
4. Import pattern and OVAs over GUI from HTTP server only; FTP, SFTP, and NFS are not currently supported.

### **Web-based user interface**

The primary access to the IBM PureApplication System is through the web-based user interface. In the web-based user interface, two consoles are available: System console and Workload console.

The System console is for administrators, superusers, and privileged users. It provides wizards to configure private clouds, manage deployed virtual machines and storage, and assist with problem determination and monitoring. By using the System console, you also can manage system resources, troubleshoot system-related issues, and diagnose problems, depending upon the privilege that is provided, as shown in Figure 3-11.

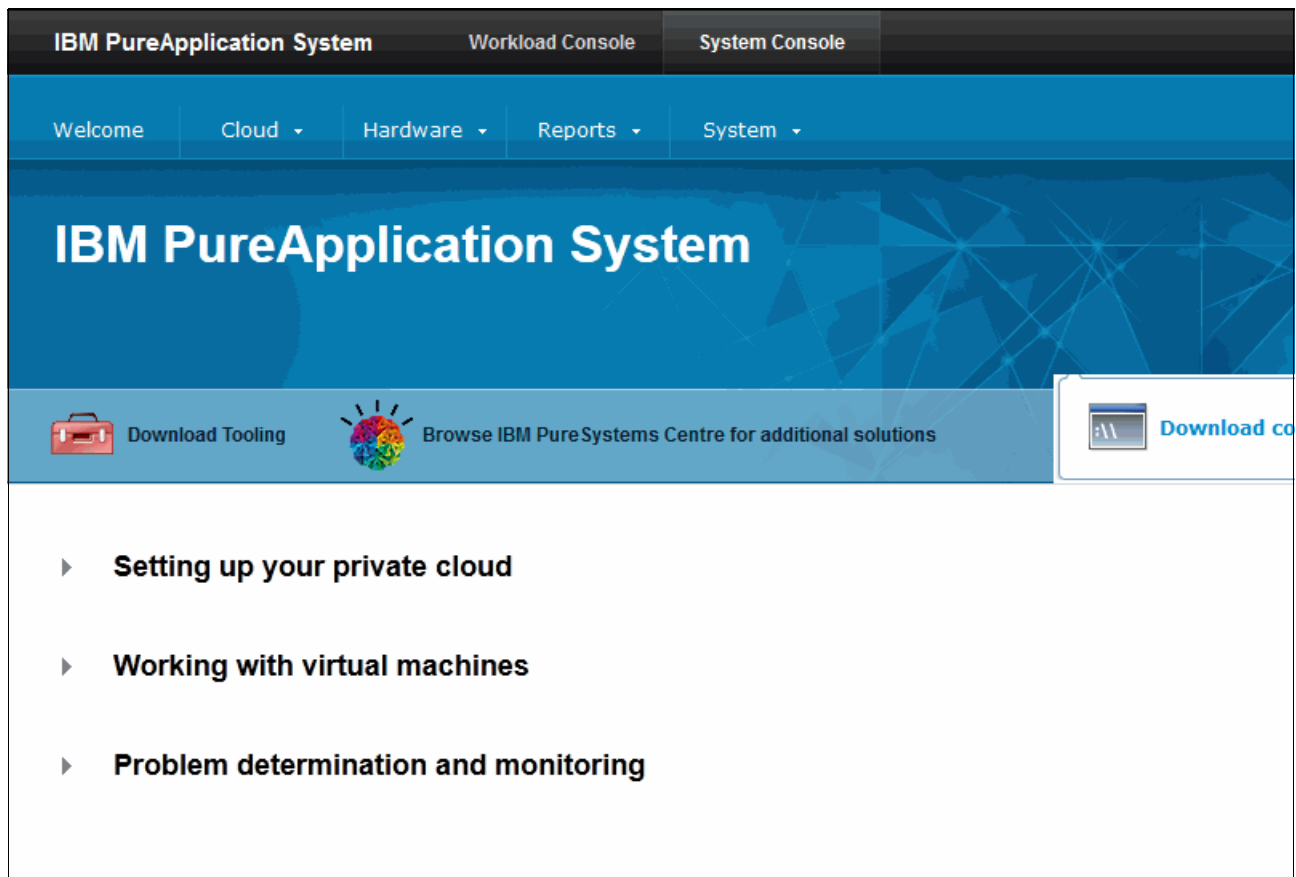


Figure 3-11 IBM PureApplication System console

The Workload console is user-specific so that they can manage, create, update, and upload their patterns, scripts, and VM. Also, users can create, deploy, and manage patterns on the cloud group for which they have access and privileges, as shown in Figure 3-12 on page 72.

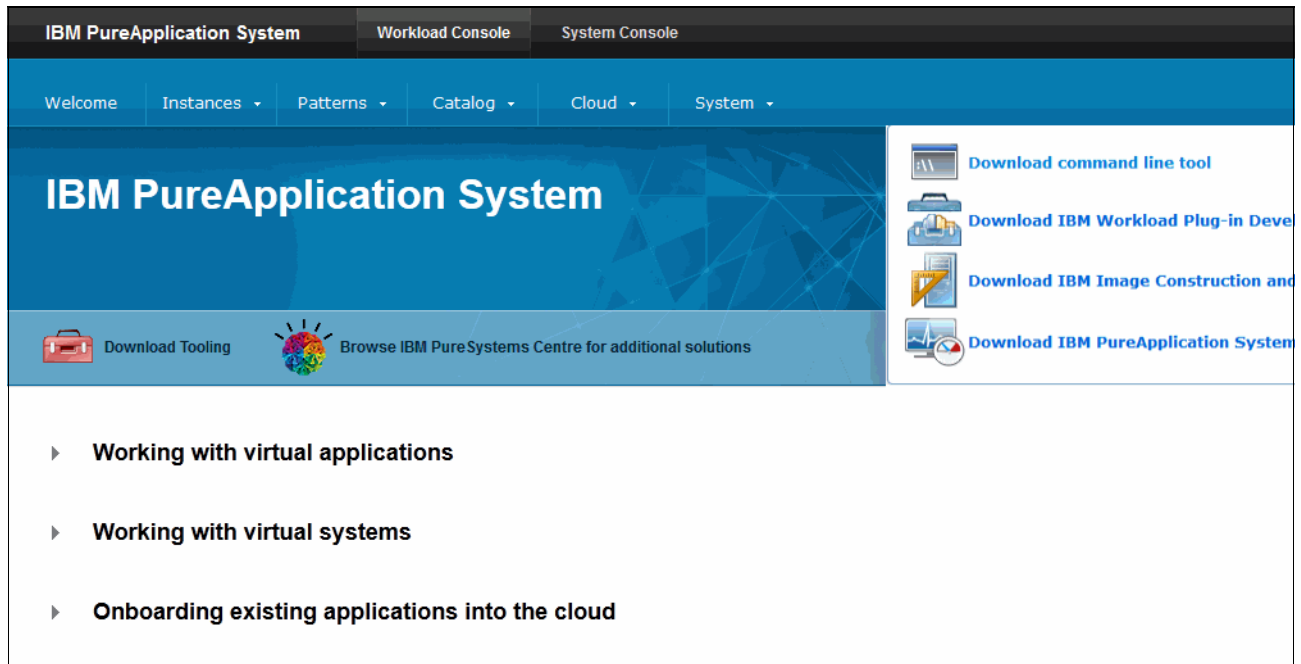


Figure 3-12 PureApplication System Workload Console

### Deploying patterns by using the web-based interface

The deployment example is an IBM SOA Policy Pattern (VSP) on a PureApplication System. On the PureApplication System, you can deploy a VSP by selecting **Workload Console** → **Patterns** → **Virtual Systems**. You see a list of VSPs on the left side of the window. Select the VSP that you want to deploy, as shown in Figure 3-13.

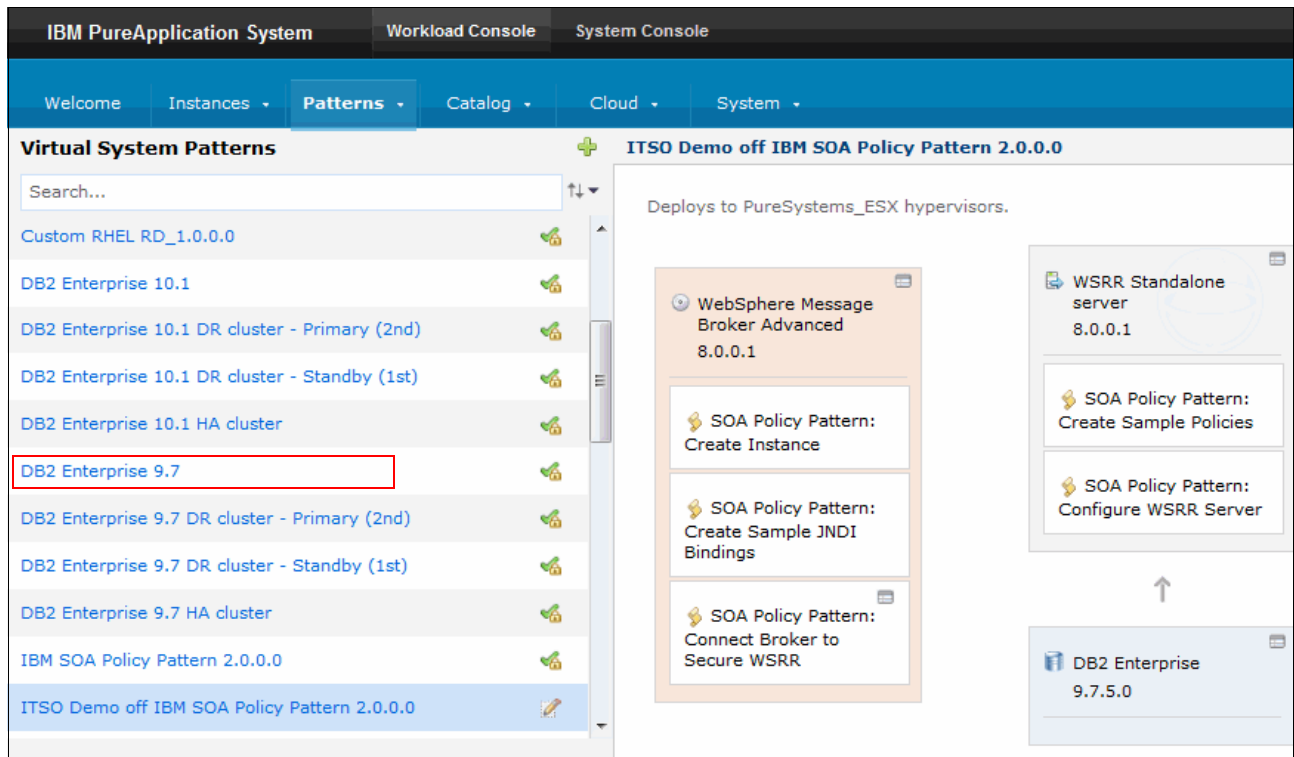


Figure 3-13 VSP Deployment on IBM PureApplication System

If you want to modify the pattern, select **Edit** in the upper right of the canvas. After you are ready to deploy, select **Deploy** in the upper right of the canvas. When you select the deploy option, window opens in which you can select the deployment environment that is displayed, as shown in Figure 3-14. Here, you can name the deploying pattern, choose the environment depending upon the access rights, schedule the deployment, and provide values to the individual part deployment variables.

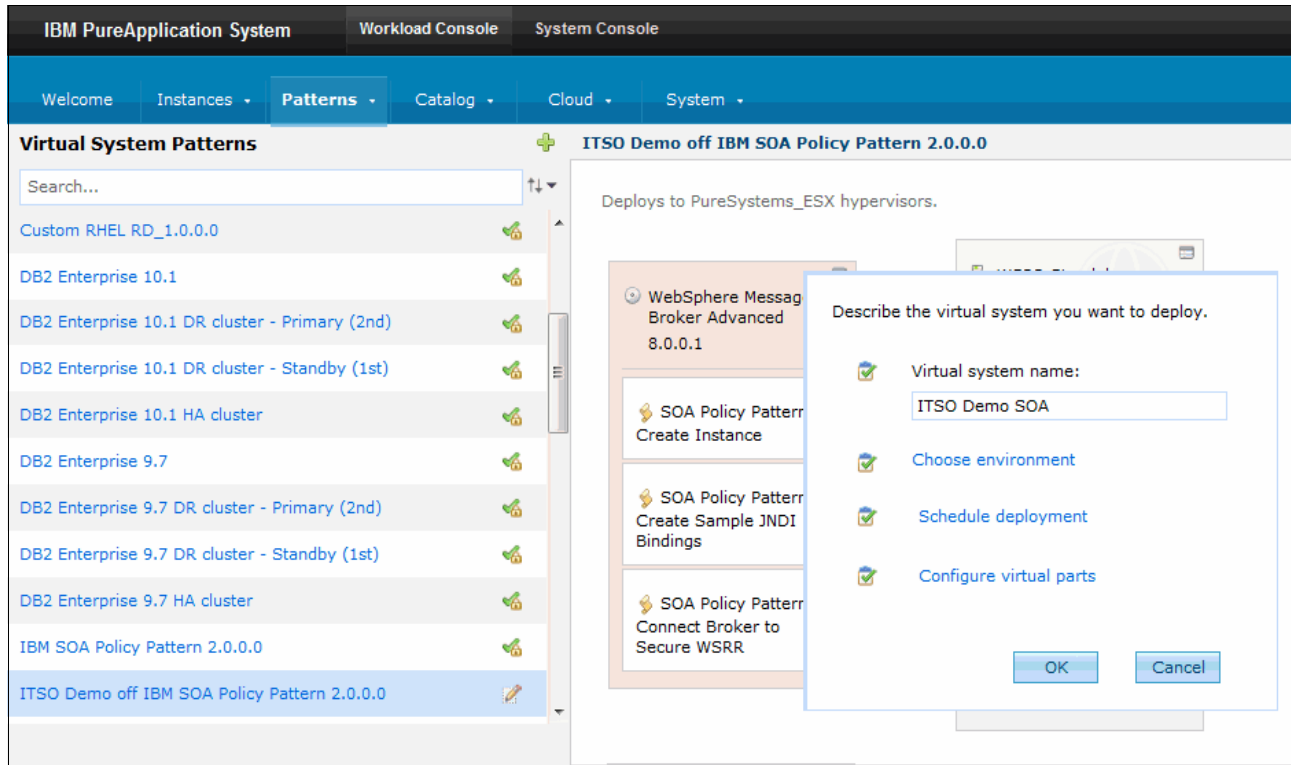


Figure 3-14 IBM SOA Pattern (VSP) Deployment on IBM PureApplication System

After you select **OK**, the system takes some time to deploy the pattern (the length of time is depends on the components that are used and the sizes of the scripts in the pattern). You can get the status of the deployment by clicking **Instances** → **Virtual Systems**. After the pattern is deployed successfully, you see a green arrow, as shown in Figure 3-15 on page 74.

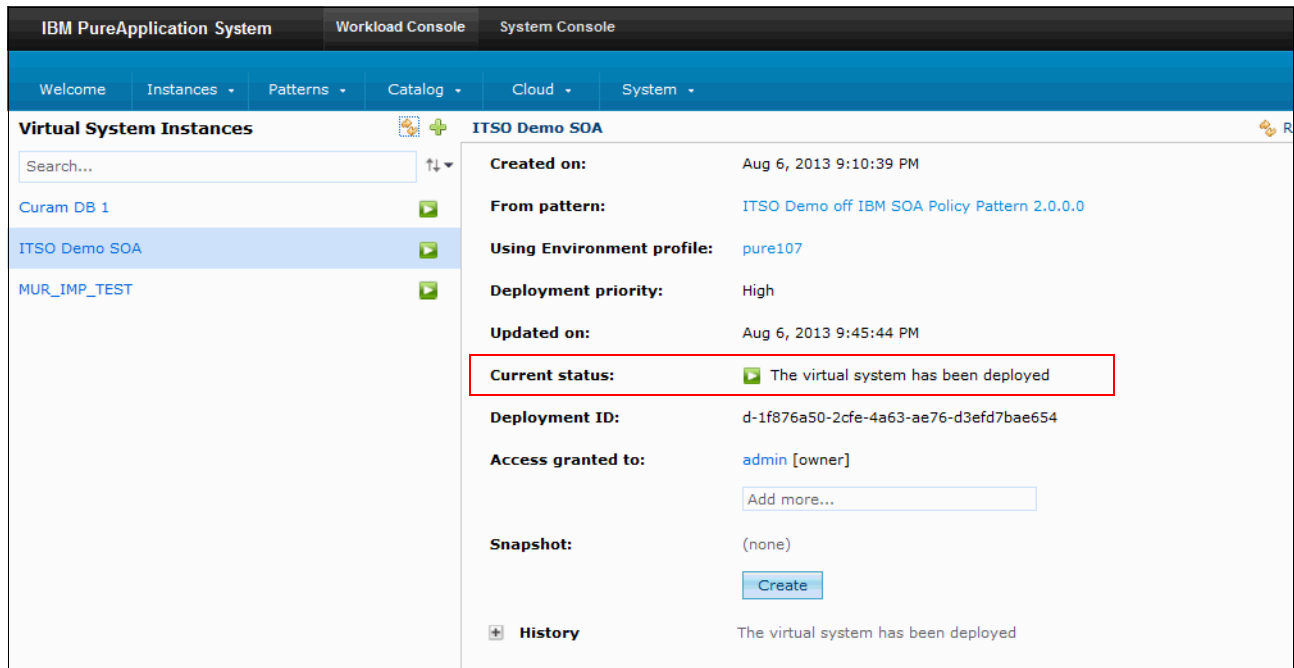


Figure 3-15 IBM SOA Pattern (VSP) Deployment on IBM PureApplication System

At any time you can stop, shut down, or start the deployed instance by selecting the respective button in the upper right corner of the window. You can also perform a system or application service fix pack management task by selecting **Service** in the upper right corner.

If you want to know the number of VMs that are deployed and their individual status, you can browse in the same page, as shown in Figure 3-16.

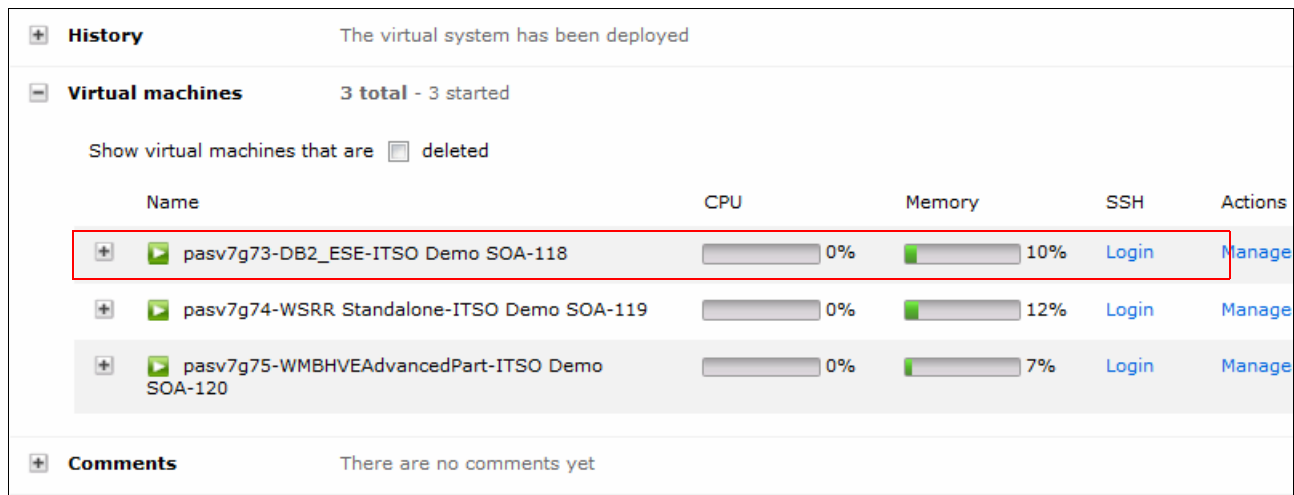


Figure 3-16 IBM SOA Pattern (VSP) deployed VMs

After you find the status of the instance and VMs, you can access the services that are deployed. There is a link to each service console that you defined in the pattern. You can access the link by expanding the individual server; in our example it is `pasv7g74.iicbang.ibm.com`. Select the link to access and log in to the service console, as shown in Figure 3-17 on page 75.



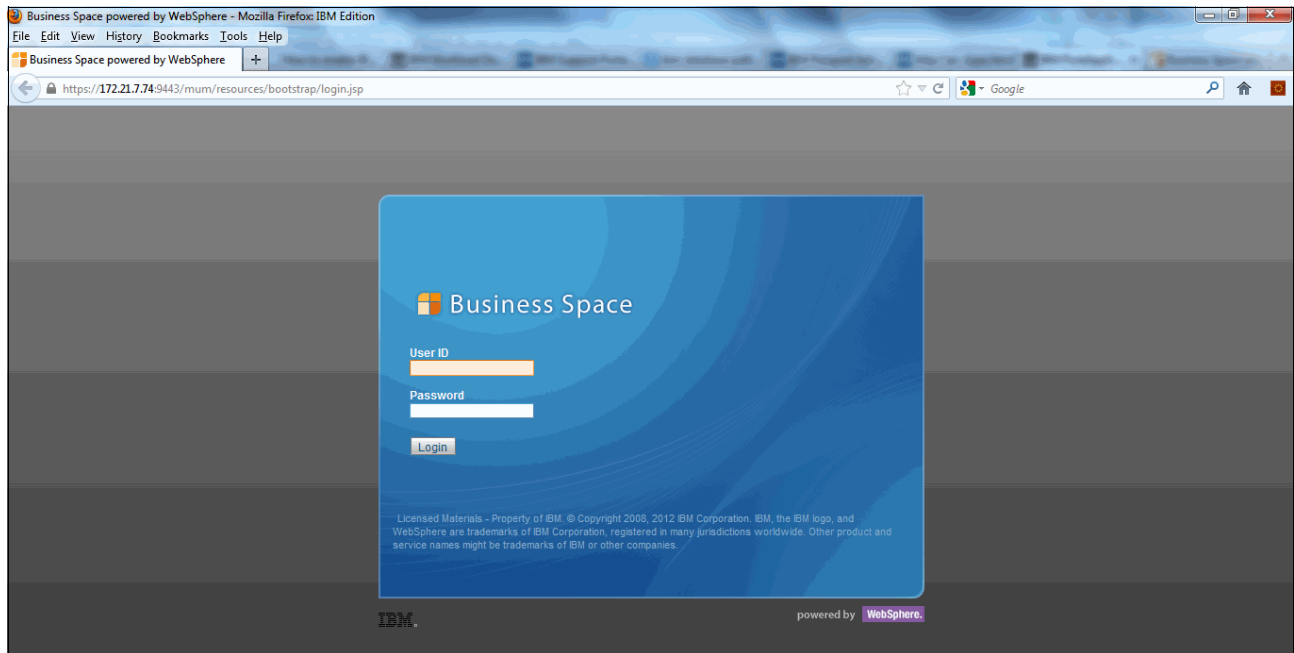


Figure 3-17 IBM SOA Pattern (VSP) Deployed Service console

## CLI

The IBM PureApplication System's CLI tool is similar to IBM Workload Deployer CLI. The CLI tool can be downloaded from the PureApplication System Workload Console interface to a Windows or Linux system. Example 3-2 shows the use of the CLI to deploy a pattern.

### Example 3-2 Deploying pattern by using command prompt

```
#./deployer -h IPAS.itso.ral.ibm.com --acceptcert -u itsouser1 -p userpasswd -f
../samples/deployPattern.py
```

It will prompt for Name for the deploying VSP

new virtual system name: *ITSO IBM SOA Pattern*

Then its going to list available pattern to deploy and wait for your input like

1. WebSphere single server
2. WebSphere cluster
3. WebSphere cluster (development)
4. WebSphere cluster (large topology)
5. WebSphere advanced cluster
6. WebSphere advanced cluster (development)
7. DB2 Enterprise 9.7
8. DB2 Enterprise 10.1
9. DB2 Enterprise 9.7 HA cluster
10. DB2 Enterprise 9.7 DR cluster - Standby (1st)
11. DB2 Enterprise 9.7 DR cluster - Primary (2nd)
12. DB2 Enterprise 10.1 HA cluster
13. DB2 Enterprise 10.1 DR cluster - Standby (1st)
14. DB2 Enterprise 10.1 DR cluster - Primary (2nd)
15. Advanced Middleware Configuration v1.0.0.3
16. BPM Advanced Clustered Pattern 8.0.0.0
17. BPM Advanced Clustered Process Center Pattern 8.0.0.0
18. BPM Advanced Clustered Process Server Pattern 8.0.0.0
19. IBM SOA Policy Pattern 2.0.0.0

*select a pattern to deploy: 19*

Then it will list the Profiles available and wait for your input like

*1. Test Profile1*

*2. Test Profile2*

*select an environment profile: 1*

Then it will prompt for scheduling the start event of the pattern like

*number of seconds to delay start (default=no delay): <Provide time and enter>*

Then it will prompt for scheduling the stop event of the pattern like

*number of seconds until stop (default=no scheduled stop): <Provide time and enter>*

then it will prompt for root and virtuser password like

*root password for virtual machines:*

*user password for virtual machines:*

once it started the instance it will provide the message like

*virtual system created:*

```
{
  "acl": (nested object),
  "created": Aug 7, 2013 10:04:24 PM,
  "currentmessage": None,
  "currentmessage_text": None,
  "currentstatus": "RM01036",
  "currentstatus_text": "Queued",
  "desiredstatus": "",
  "desiredstatus_text": "",
  "environmentprofile": (nested object),
  "id": 68,
  "maintenances": (nested object),
  "name": "ITSO IBM SOA Pattern",
  "owner": (nested object),
  "pattern": (nested object),
  "priority": 4,
  "snapshots": (nested object),
  "updated": Aug 7, 2013 14:04:24 PM,
  "virtualmachines": (nested object)
}
```

#

---

### 3.3.4 SmartCloud Application Service

IBM SmartCloud Application Services is an IBM Platform as a Service (PaaS) that powers fast development and deployment of applications to the cloud with the following suite of services and components:

- ▶ Cloud-based development tools
- ▶ Workload patterns
- ▶ Middleware
- ▶ Databases

IBM SmartCloud Application Services (SCAS) runs on the IBM self-service public cloud, SmartCloud Enterprise.

The purpose of IBM SCAS is to provide end-to-end services for applications from development to production. SCAS consists of the following areas:

- ▶ Application lifecycle
- ▶ Application resources
- ▶ Application environments
- ▶ Application management
- ▶ Integration

IBM SCAS features the following main services that cover the application lifecycle:

- ▶ IBM SmartCloud Application Collaborative Lifecycle Management Service with which you can manage your application lifecycle development. It is based on Rational products, including the following products:
  - Rational Team Concert
  - Rational Quality Management
  - Rational Requirement Composer
- ▶ IBM SmartCloud Application Workload Service is software that provides access to virtual images and patterns, which can be used as they are or customized and then used. The topology then can be securely deployed on the IBM SmartCloud Enterprise via virtual application or system patterns that are managed and maintained in a public cloud.

The IBM SmartCloud Application Collaborative Lifecycle Management Service and IBM SmartCloud Application Workload Service are used for the DevOps and Production scenario together. For example, in Rational Application Developer, a developer tracks the defect that is managed by the IBM SmartCloud Collaborative Lifecycle Management Service and uses the Workload Deployer plug-in to create a test environment on the IBM SmartCloud Enterprise by using the IBM SmartCloud Application Workload Service. After the application is ready, the application can then be deployed for production on same IBM SmartCloud Application Workload Service and the updates and fix packs can be managed.

**Note:** In this book, only IBM SmartCloud Application Workload Service for pattern usage is described.

## IBM SmartCloud Application Workload Service

The SmartCloud Application Workload Service provides a run time and management service to help accelerate the creation, deployment, and management of application workloads without the need to manage the details of the middleware and infrastructure. The service provides policy-based automated scaling and management of the application.

IBM SmartCloud Application Workload Service is a feature of IBM Workload Deployer, which is the brains of the workload service. It uses workload patterns (VSP and VAP) to create consistent, validated, and repeatable virtual applications within the cloud environment. Workload patterns are solutions that enable the deployment of complex business applications in a cloud environment. When you use a workload pattern, you can focus on an application instead of the middleware infrastructure in which the application runs.

For example, if an application consists of a WebSphere Application Server application, a database schema, and an LDAP program to manage users, you deploy these applications. SmartCloud Application Workload Service installs and configures the required middleware and manages the application run time by using policies that you define.

Cloud computing is a computing paradigm in which data and services are in data centers. The data and services can then be accessed from any connected devices over the Internet. Applications can use the cloud for added value, such as storage, queuing, and hosted applications. The applications also can be hosted on the cloud.

Application Workload Service includes the following features:

- ▶ Support for Java apps, PHP (other languages and applications, such as, .NET and Ruby)
- ▶ Web Application Services (Web Application Pattern that is based on WebSphere Application Server, auto scaling that uses caching and proxy, and auto-failover and restart)
- ▶ Virtual databases
- ▶ Virtual System Patterns
- ▶ Virtual Application Patterns

### ***Considerations for IBM SmartCloud Application Workload Service***

Before you begin, you must have a SmartCloud Enterprise user ID. The SmartCloud Enterprise user ID is required to create and start a SCAWS Instance. Later, you can create user IDs for users who are using the SmartCloud Application Workload Service instance only; for those users, a SmartCloud Enterprise user ID is not required.

Assuming that all of the required SmartCloud Enterprise resources are available, complete the following steps to start the IBM SmartCloud Application Workload Service on SmartCloud Enterprise:

1. In the SmartCloud Enterprise console, click **Control panel** → **Service Instances**.
2. Click **Add a service instance to get started**. A four-step wizard for creating a SmartCloud Application Workload Service instance opens:
  - a. In step 1 of the wizard, from the Type list, select **Public**. From the Data Center list, select the required data center for hosting your service. Click the right arrow icon to refresh the list of available service offerings.

Depending on the number of virtual machines to be deployed, select the required service offering with the SmartCloud Application Workload Service that is indicated in the name (large, medium, small). Click **Next**.
  - b. In step 2 of the wizard, enter a name and a short description for the SmartCloud Application Workload Service instance and then click **Next**.
  - c. In step 2a, enter the following information about the SmartCloud Application Workload Service instance:
    - Administrator Password
    - Reenter password
    - Persistent Storage ID
    - Full host name
    - Service IP addressClick **Next**.
  - d. In step 3, verify the information that was entered and then click **Next**.
  - e. In step 4, review the service agreement and then click **I agree**.
3. Click **Submit**. A confirmation message “Service instance request SCAWS Instance Name has been successfully submitted” is displayed.
4. Click **Return to control panel**.

You now have a SmartCloud Application Workload Service instance and a service instance is listed in the control panel. You can then connect to instance by using its IP address. The default user ID for the instance is `cbadmin` and password is the one that you defined during the instance setup process.

After you log in, you see the web interface, as shown in Figure 3-18. Similar to other SmartCloud platforms, you can interact with the SmartCloud Application Workload Service instance by using the web-based Interface, CLI, or REST API.

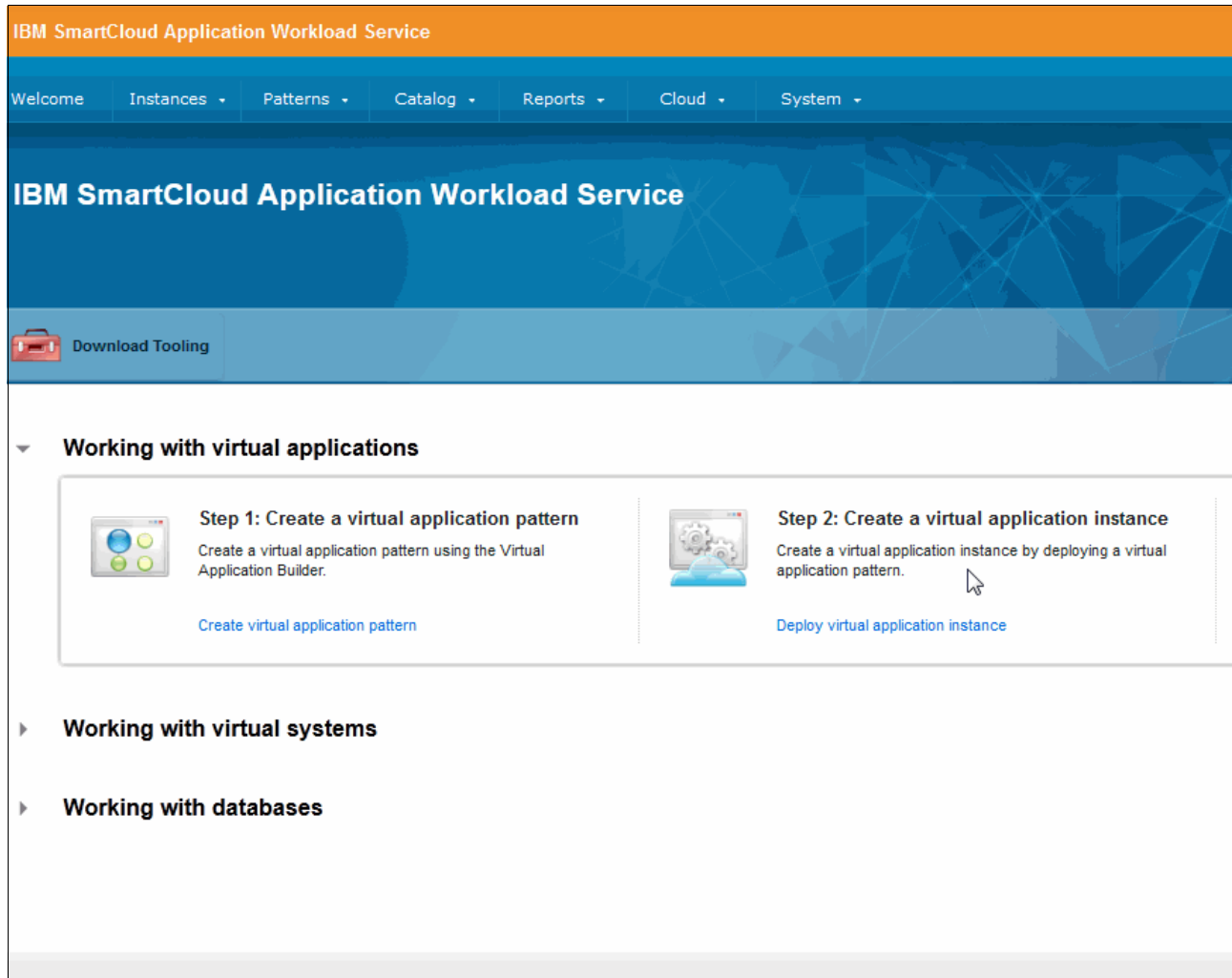


Figure 3-18 SmartCloud Application Workload Service web-based interface

### ***Deploying a pattern on SmartCloud Application Workload Service***

The IBM Worklight Pattern (virtual application pattern) with sample application is used for the deployment example. On the SmartCloud Application Workload Service console, you can deploy the pattern by selecting **Patterns** → **Virtual Application Patterns**.

On the canvas, you see a down arrow menu. Select **IBM Mobile Application Pattern Type 6.0**. On the left side of the window, there is list of patterns. Select the pattern **ITSO Mobile Pattern**, as shown in Figure 3-19 on page 80.

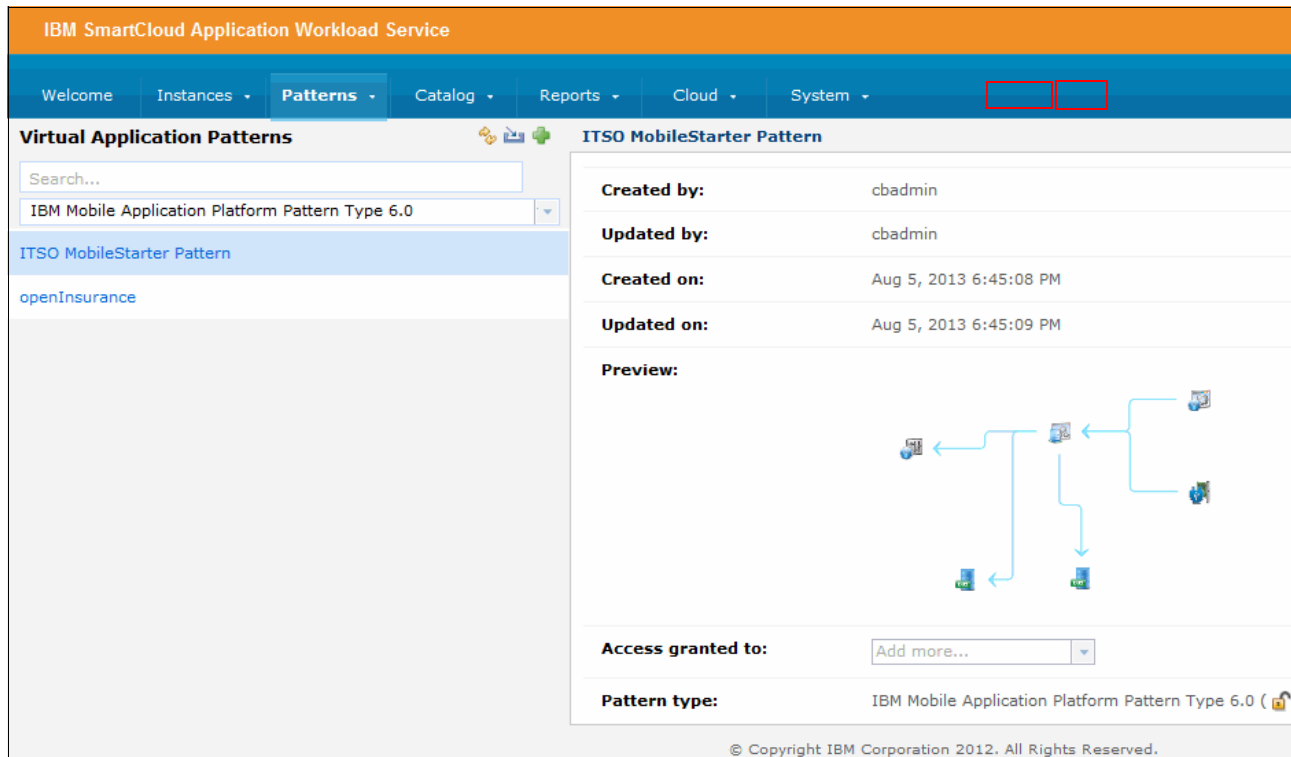


Figure 3-19 Mobile Application Pattern Deployment on SmartCloud Application Workload Service

To modify the pattern, select **Open** in the upper right of the canvas. A new tab opens in the browser editing canvas for the application pattern. After you complete editing, save and then close that tab. When you are ready to deploy the pattern, click **Deploy** in the upper right of the canvas. When you select the deploy option, a window opens in which information for the name for the pattern, the deployment environment, and billing model (each instance size, software licensing) is available, as shown in Figure 3-20. You can add a name to the deploying pattern, choose the environment depending upon the requirements, schedule the deployment, and configure the individual part's deployment variables.

Deploy Virtual Application

Name:

IP Version: ☒ IPv4 ☐ IPv6

Cloud group:

☐ Estimate cost of deployment:

Pattern	# Units	Tier	License	Billing	Unit Price
Transactional DB (WLRuntimeDB-db2)	1	Small	BYOL	Hourly	0.64 USD /UHR
Transactional DB (WI ReportsDB-db2)	1	Small	BYOL	Hourly	0.64 USD

☐ Advanced

Figure 3-20 SmartCloud Application Workload Service Deploy Virtual Application options

After you select **OK**, the system takes time to deploy the pattern. You can get the status of the deployment by clicking **Instances** → **Virtual Application Instances**. After the pattern is deployed successfully, you see a green arrow, as shown in Figure 3-21.

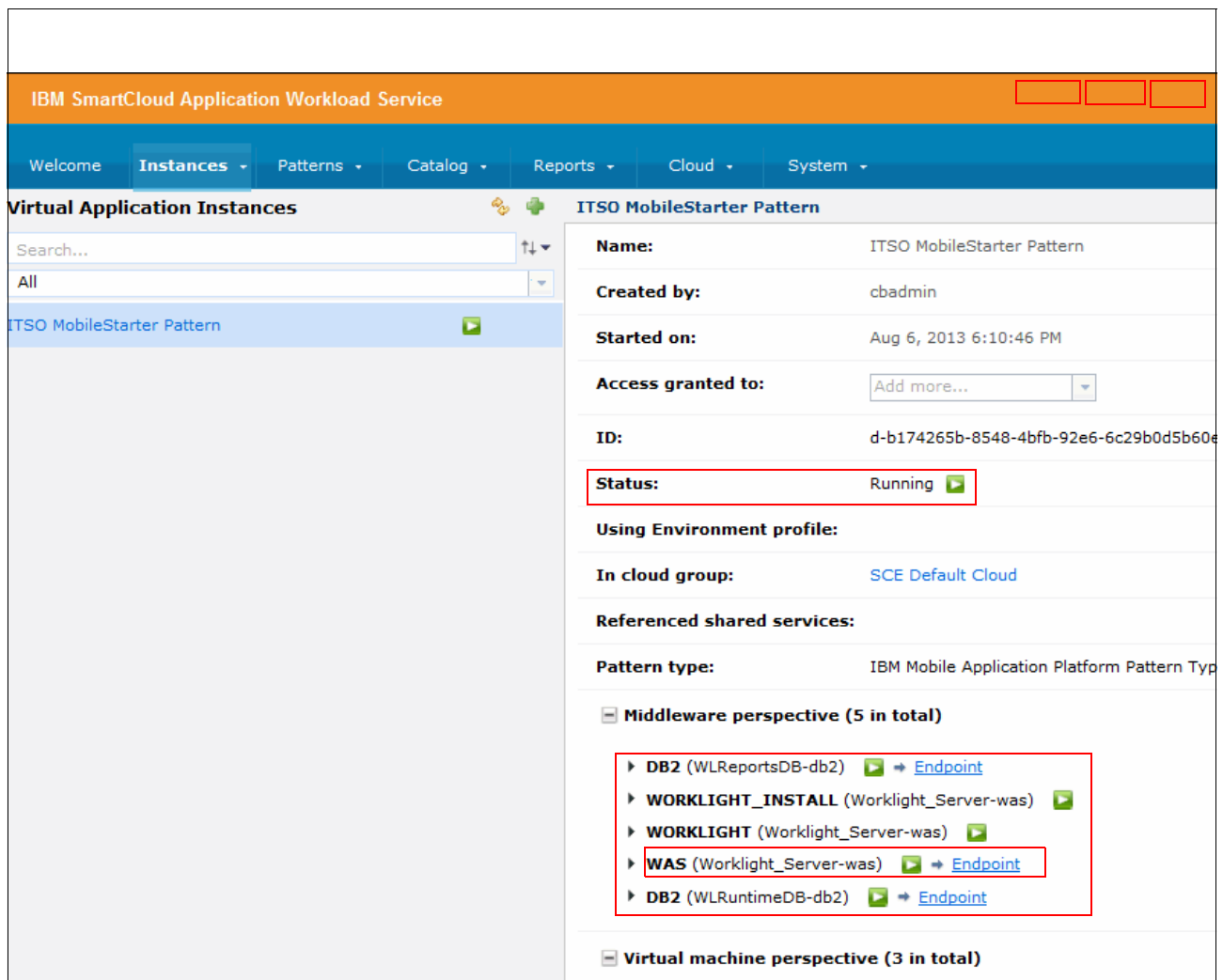


Figure 3-21 Worklight Pattern Deployment Status on SmartCloud Application Workload Service

You can delete, upgrade, and manage the deployed instance by selecting the required button in the upper right corner of the window. You can get the application endpoint URL by clicking **Endpoint** next to the application server.

### 3.4 Deployment solutions made easy with patterns

A pattern is a model of a multi-server runtime environment, which is represented in a file. It can be interpreted by a deployment platform and shared between users and teams. By using patterns, you can build highly usable solutions that incorporate many integrated virtual machines, software components, and configuration elements. By using patterns, you can deliver complex solutions in a single, deployable unit across IBM SmartCloud Platforms, as shown in Figure 3-22 on page 82.

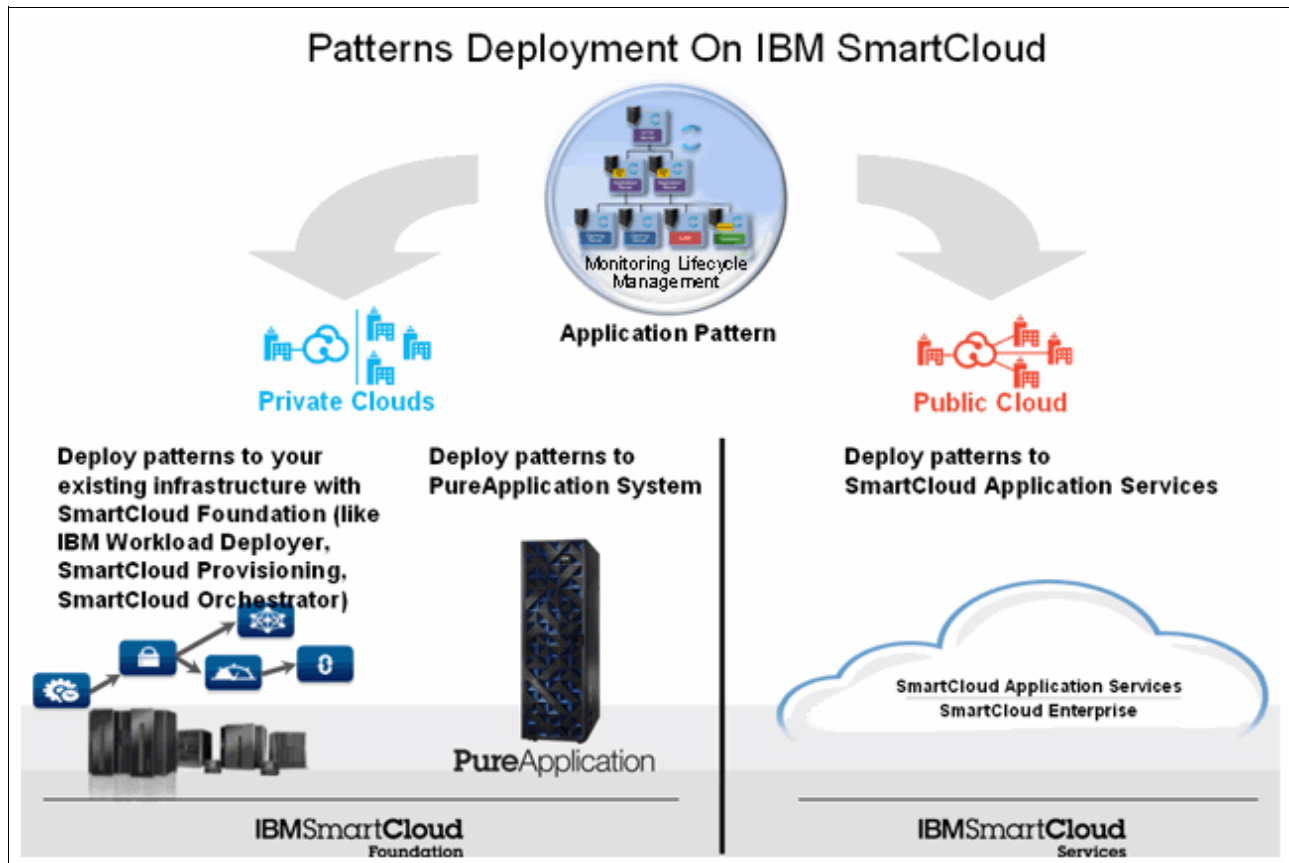


Figure 3-22 Pattern Deployment Platforms

When you are importing any new pattern from an external source to your system, you must have prerequisite and dependency checked in the catalog. For example, for a virtual application pattern to use the Worklight Application Pattern requires Foundation pattern 1.0.0.6 and the web application pattern 2.0.0.5 (both are downloadable from IBM Support: Fix Central, which is available at this website:

<https://www-933.ibm.com/support/fixcentral/>

If you do not have the required dependent patterns and versions on the system, you can import the worklight pattern and plug-in-type, but you cannot deploy and use the pattern.

Complete the following steps:

1. Refer to the prerequisite documents for the worklight pattern, which are available at this website:  
<http://www.ibm.com/developerworks/mobile/worklight/>
2. Check for the existence of prerequisite on the system. If none exist, download and import the dependency patterns to the system and enable the patterns.
3. Import the worklight pattern and pattern-type.
4. Create and customize the pattern.
5. Deploy the pattern.



Importing the Virtual System Pattern can fail for the following reasons:

- ▶ The OVA and Hypervisor edition that the pattern requires might not be available on the system.

If the name of the hypervisor is wrong, it can be fixed by editing the JSON file. Also, before the VSP is imported, the correct version of the middleware or OVA that is required by the VSP must be available.

- ▶ Not enough disk space.
- ▶ Security and access limitations.
- ▶ Referring location problem.





## Pattern lifecycle

In the broadest sense, managing the lifecycle of your IT infrastructure is about repeating the deployment and configuration of applications and middleware as they move through different stages of operation and existence. Patterns of expertise help in this context by speeding up deployment activities and more consistently replicate the configuration of the environment. The goal is to ensure that the way an application behaves during testing is the way that it behaves in production.

More narrowly, each part of a system, be it an application, a middleware component, an operating system, or a resource, has specific lifecycle stages of its own. The focus of this chapter is the post-deployment management of a pattern instance and managing the lifecycle of the pattern, whether it is a virtual system pattern or a virtual application pattern.

This chapter includes the following topics:

- ▶ Overview
- ▶ Virtual system lifecycle
- ▶ Virtual application lifecycle
- ▶ Red Hat OS Update Service
- ▶ Monitoring deployed instances
- ▶ Pattern management and continuous delivery

## 4.1 Overview

A pattern-based virtual system lifecycle can be broken down into the following high-level phases:

1. Dispense a pattern to the cloud to reserve the necessary resources (IP addresses, storage, processors, memory, and so on) and create the system instance.
2. Run the system instance when you need it.
3. Remove the resource reservation by deleting or storing the system (storing a system releases the hardware resources, but not the IP addresses).
4. Delete the system instance to return all resources to the resource pools.

These system lifecycle phases are shown in Figure 4-1.

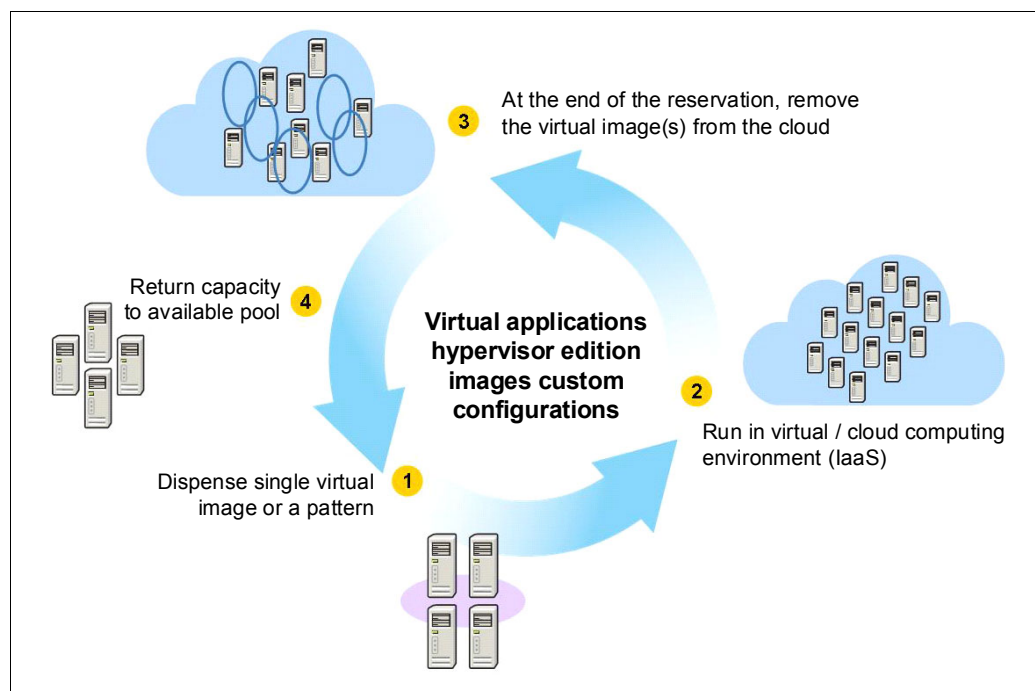


Figure 4-1 Lifecycle of a deployed system

Virtual system and virtual application patterns are composed of smaller components or parts.

- ▶ A virtual system pattern (VSP) can be composed of, for example, WebSphere Application Server Hypervisor Edition images (provided by IBM or custom-built), script packages, and the pattern definition. Each Hypervisor Edition image is composed of an operating system and software that is installed on it.
- ▶ A virtual application pattern (VAP) can be composed of a base operating system image and the workload that is deployed on it. At a basic level, the workload consists of software packages and scripts.

This means that managing the lifecycle of a pattern involves not only the lifecycle of the deployed instance (a virtual system or a virtual application instance), but also the lifecycles of the individual pattern components.

Additionally, because a pattern is often deployed to different application environments (test, pre-production, production, and so on), if you import a new version of the pattern, you sometimes must upgrade the deployed instances. The evolution of a pattern can involve upgrading a single component or a combination of components.

In the next two sections, we describe the effect of a pattern change on a deployed pattern instance. We then describe the lifecycle management of a pattern.

## 4.2 Virtual system lifecycle

At a high level, a Virtual System Pattern is built with the following components:

- ▶ Hypervisor Edition image (operating system and preinstalled software)
- ▶ Script package
- ▶ Pattern topology

From a lifecycle perspective, as a pattern evolves (whether it is for a software upgrade, configuration change, or something else), changes to any of its components must be applied to all instances of the pattern that already are deployed.

IBM provides a large set of Hypervisor Edition images, each consisting of an operating system and preinstalled and pre-configured middleware. The following images are available:

- ▶ IBM WebSphere Application Server
- ▶ IBM DB2
- ▶ IBM WebSphere Portal
- ▶ IBM Informix
- ▶ IBM Domino®

You can customize these Hypervisor Edition images by adding needed components (you can add or remove packages that are based on your operating system security requirements or add software). You also can build your own Hypervisor Edition images.

Customizations can be made at the operating system, middleware, or application levels. The following examples of potential customizations are shown in Figure 4-2 on page 88:

- ▶ An IBM Hypervisor Edition image can be extended by the infrastructure team to create a custom image that meets a specific need, such as through the installation of third-party software.
- ▶ A pattern can be customized by the infrastructure team to include components that are needed by an application but are not included in the default pattern.
- ▶ A customized pattern can be further customized by the application team, such as by installing an application.
- ▶ The deployment team can customize a pattern to provide parameters that are specific to the system that is being deployed.

These layers of customization typically relate to the different teams (infrastructure, operations, and application) that are in charge of maintaining the various pieces of the system.

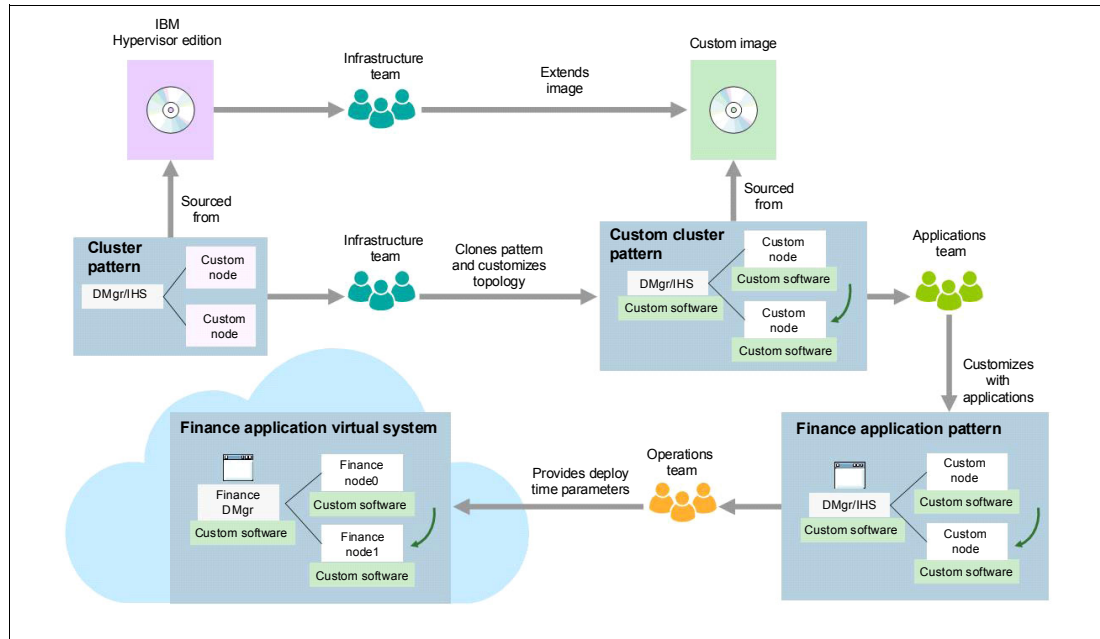


Figure 4-2 Customizations starting from a standard Hypervisor Edition image and a standard pattern

In the following sections, the pattern lifecycle and the customizations that can be performed on various layers of the deployment infrastructure are described.

## 4.2.1 Customizing Hypervisor Edition images

If you must use a non IBM product in your virtual system or an IBM product that is not provided in the Hypervisor Edition format, you can create your own custom image or extend a Hypervisor Edition image to add the needed software.

### Using the extend and capture approach

The option to extend an image and capture it for future use should be considered when you must customize an image.

The main alternative to extending and capturing an image is to use script packages, which are described in “Using a script package” on page 96. However, you should choose the extend and capture approach when the customization activity is time-consuming and you need the customization when you use the particular Hypervisor Edition image. An example is the installation of a monitoring agent. If you use a script package to complete this task, the agent is installed when you deploy the image, which can delay the VSP deployment. However, by using the extend and capture approach, the installation is done only once and the software is then available in the extended image whenever you deploy it.

To extend and capture an image, complete the following steps from the IBM Workload Deployer console:

1. Select **Catalog** → **Virtual Images** (see Figure 4-3 on page 89) to open the Virtual Images catalog in which you find a list of available images.

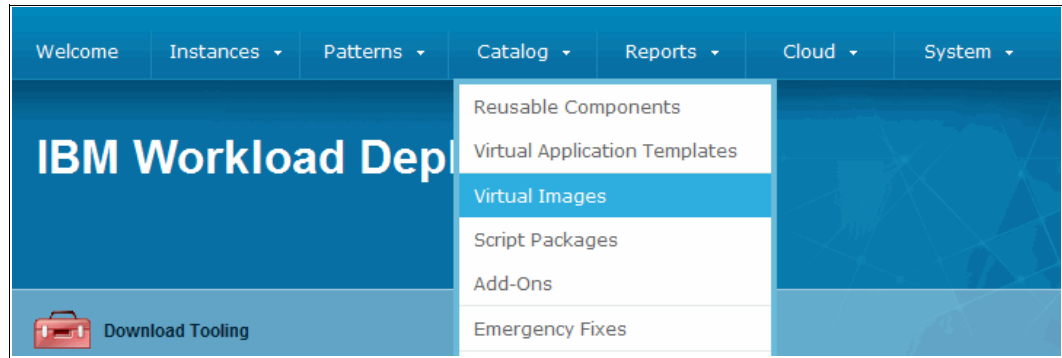


Figure 4-3 Accessing the Virtual Images catalog

2. Choose an image from the list to extend. In this example, the authors chose the IBM Business Process Manager Advanced 8.0.1.0 RHEL 6 x64 image.
3. Click **Extend**, as shown in Figure 4-4. The extension process starts.

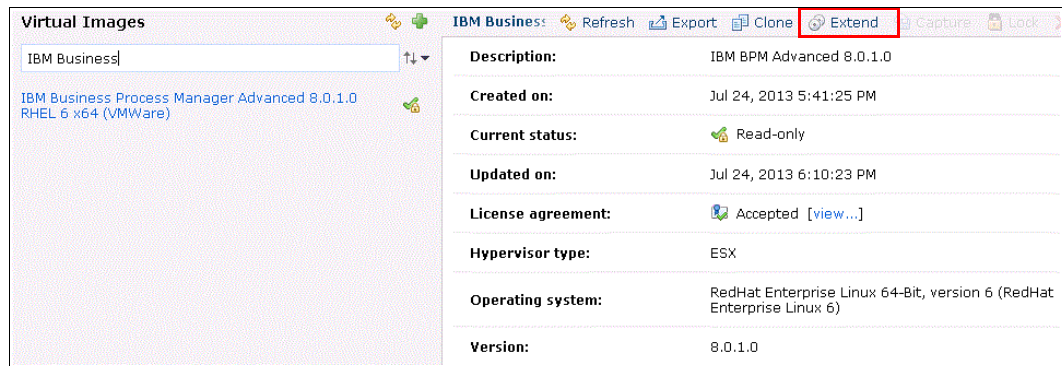


Figure 4-4 Selecting the Extend option

4. Enter a name, description, and version number for the extended image instance, as shown in Figure 4-5.

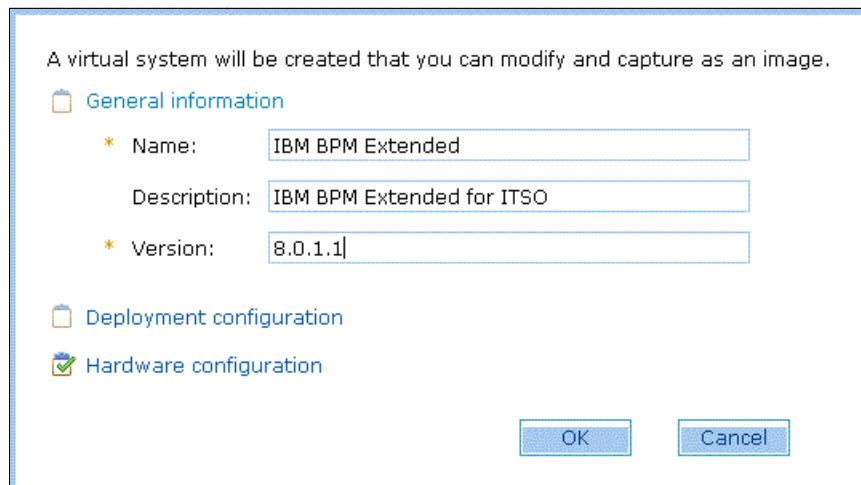


Figure 4-5 Identifying the new, extended image

5. Customize the new image as necessary by adding or removing configurations or software.

For more information about image customization, see Chapter 6 of *Virtualization with IBM Workload Deployer: Designing and Deploying Virtual Systems*, SG24-7967, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg247967.html?open>

One specific example of customization involves changing the hardware characteristics of the selected Hypervisor Edition image by increasing the disk size, as shown in Figure 4-6.

A virtual system will be created that you can modify and capture as an image.

- General information
- Deployment configuration
- Hardware configuration
  - \* Network interfaces: 1
  - \* RHEL62-64.vmdk (GB): 12
  - \* BPM\_Binaries.vmdk (GB): 20

OK Cancel

Figure 4-6 Changing the hardware characteristics of the extended image

**Note:** Depending on the technology you are using, you might have fewer customization options than are shown in the Workload Deployer images that are used in this chapter. For example, IBM PureApplication System Version 1.1 does not offer the option to increase the disk size of the images.

6. After you complete the extension process, capture the image by clicking **Capture**, as shown in Figure 4-7.

Patterns Catalog Reports Cloud System

IBM BPM Ext Refresh Export Clone Extend **Capture** Lock Delete

Description:	IBM BPM Extended for ITSO
Created on:	Aug 9, 2013 4:04:34 AM
Current status:	Draft
Updated on:	Aug 9, 2013 4:48:07 AM
License agreement:	Accepted
Hypervisor type:	ESX
Operating system:	RedHat Enterprise Linux 64-Bit, version 6 (RedHat Enterprise Linux 6)

Figure 4-7 Capturing the extended image



This action makes the extended image available in the catalog for future use, as shown in Figure 4-8.

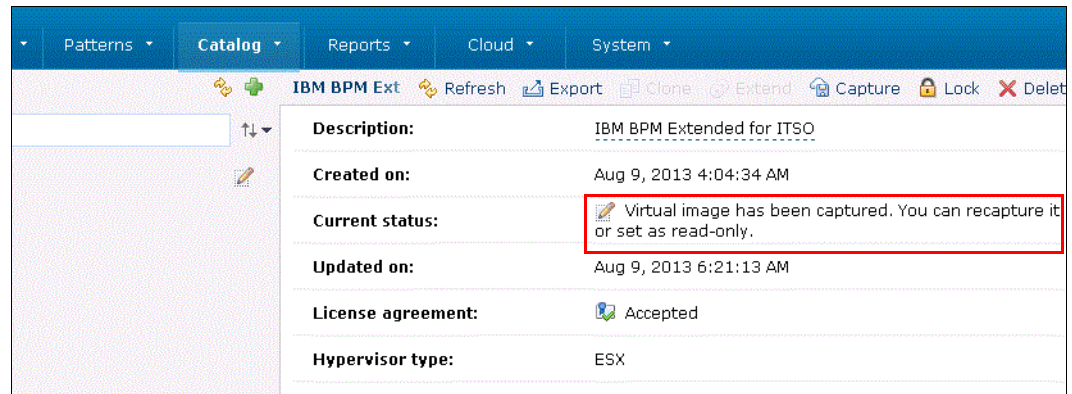


Figure 4-8 Catalog listing for new, extended image

### Creating a custom image

If extending a Hypervisor Edition image does not work for you, you can create a custom virtual image by using the IBM Image Construction and Composition Tool (ICCT).

Users of Workload Deployer or PureApplication System can download the ICCT by clicking **Download IBM Image Construction and Composition Tool** in the Welcome window, as shown in Figure 4-9.

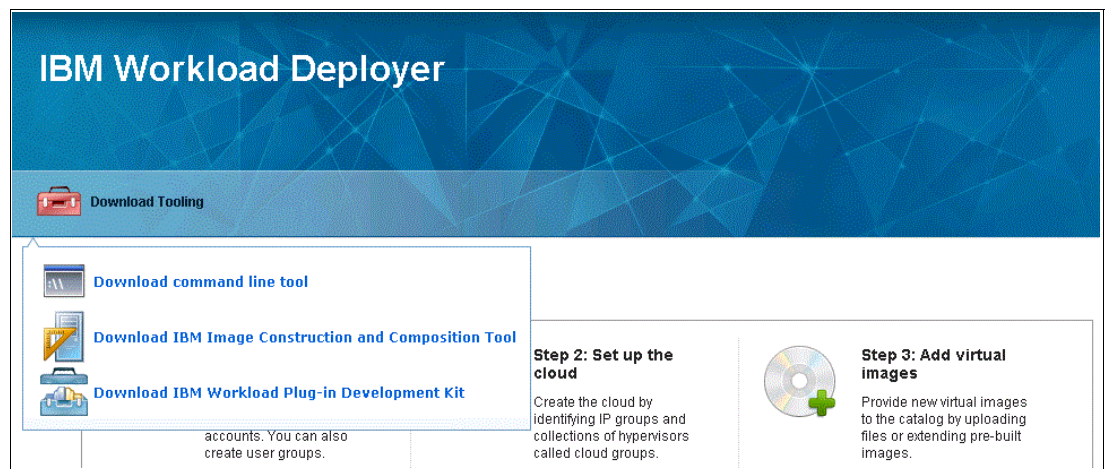


Figure 4-9 IBM Image Construction and Composition Tool download link

Instructions for installing ICCT are available in the IBM Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2FICON%2Ftopics%2Fwd\\_cicn\\_installing.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2FICON%2Ftopics%2Fwd_cicn_installing.html)

Users of PureApplication System have the option of using the ICCT virtual application pattern. This method gives you the benefits of the virtual application approach but frees you from having to install and setup ICCT. Figure 4-10 shows the ICCT pattern type in PureApplication System.

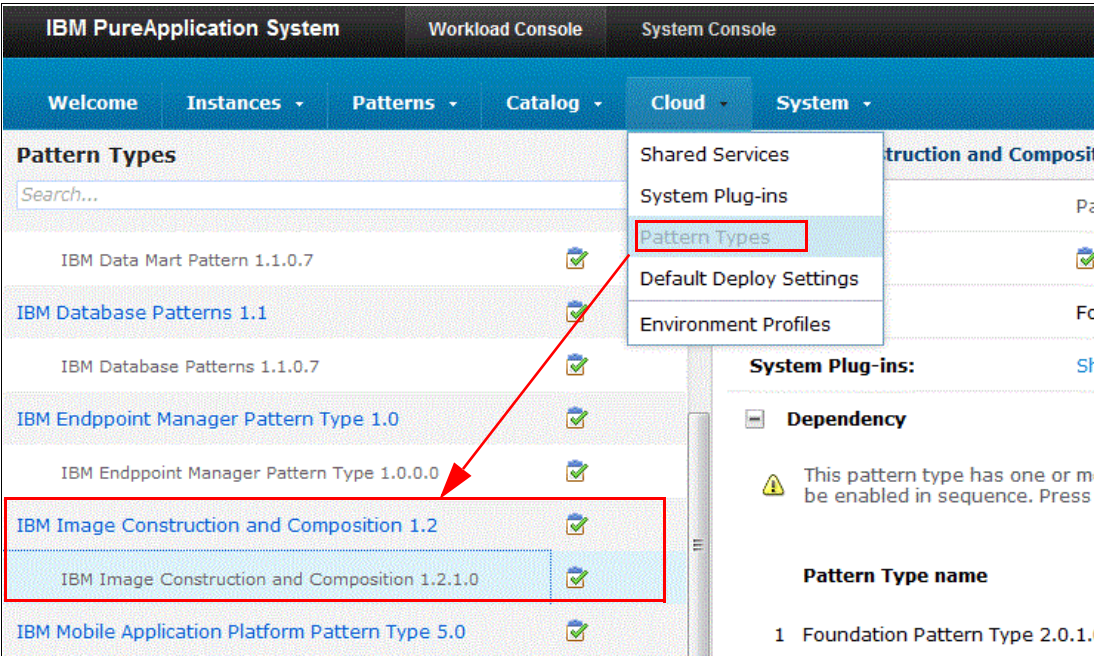


Figure 4-10 Accessing the ICCT virtual application pattern type

Complete the following steps to deploy the ICCT VAP on PureApplication System:

1. Click **Patterns** → **Virtual Application Pattern** and then select **IBM Image Construction and Composition Tool 1.2** from the drop-down menu, as shown in Figure 4-11.

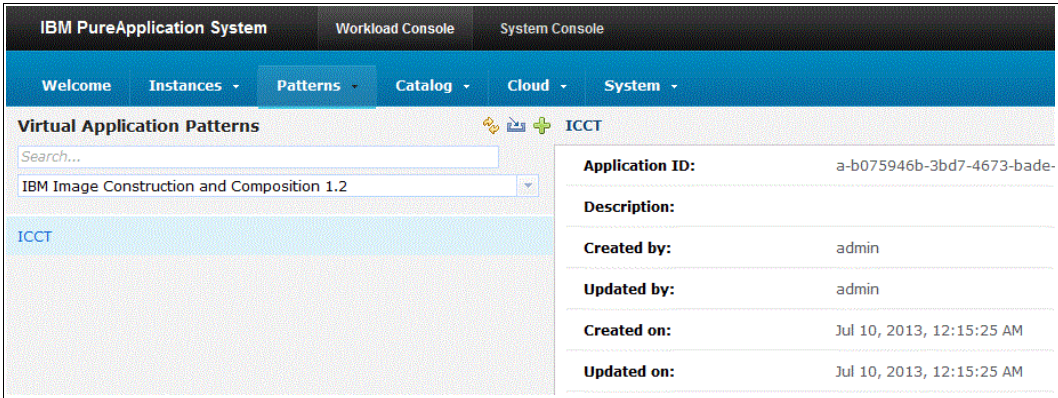


Figure 4-11 Selecting ICCT for deployment

- Click **Patterns** → **Virtual Application Pattern**, select the ICCT virtual application pattern, and then click **Deploy**, as shown in Figure 4-12.

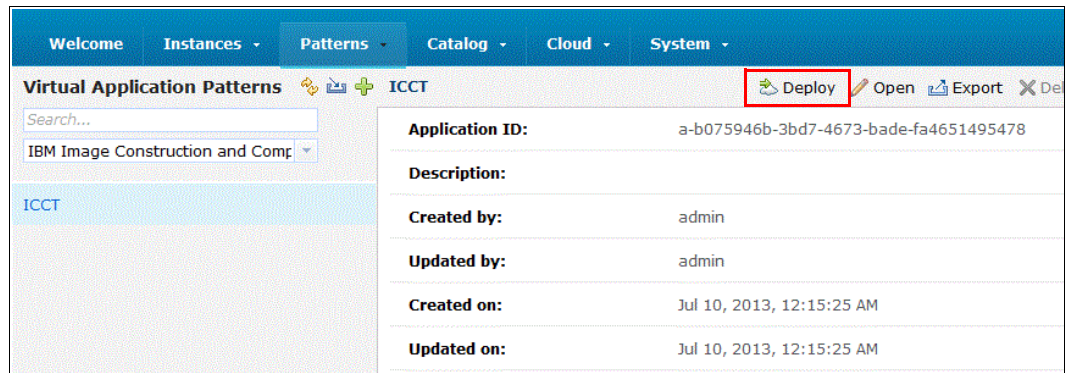


Figure 4-12 Deploying the ICCT VAP on PureApplication System

- ICCT starts automatically. After the virtual application is deployed, you can directly access the ICCT console by going to the virtual application instance page and clicking **Endpoint**, as shown in Figure 4-13.

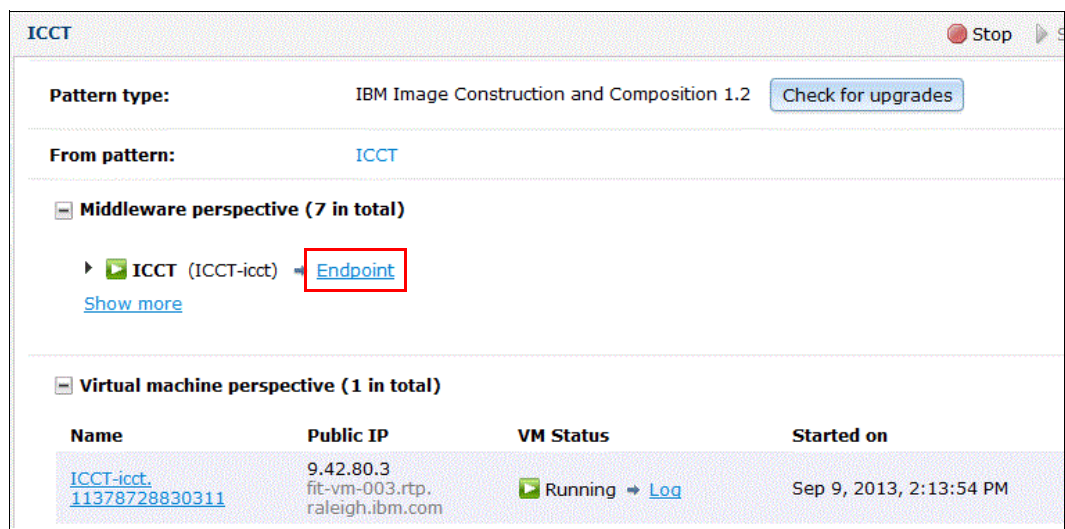


Figure 4-13 Endpoint hyperlink to access the ICCT console

4. From the ICCT console, you can build your own image, as shown in Figure 4-14.

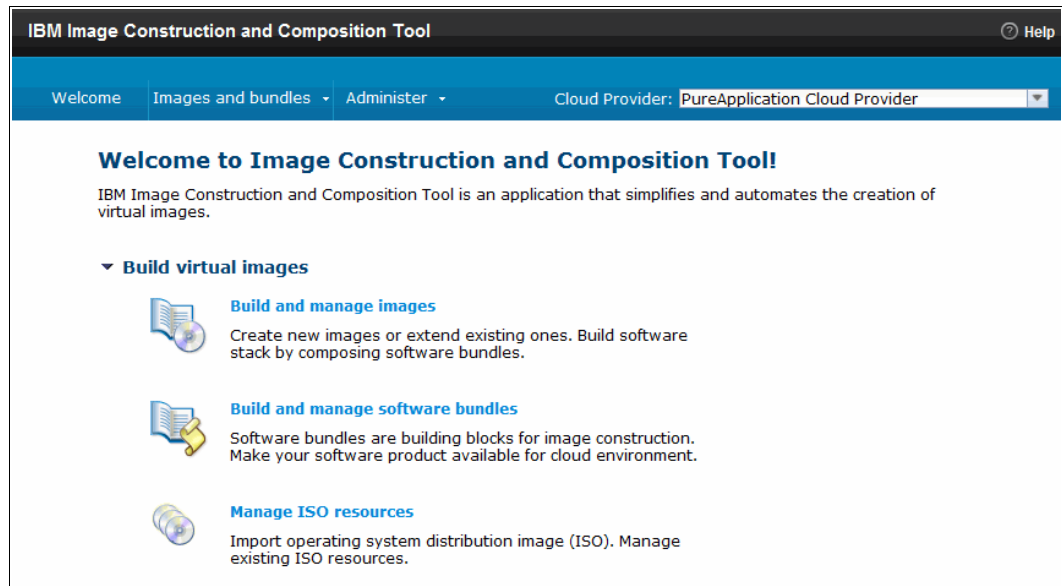


Figure 4-14 ICCT console

When you are using ICCT, you work with the following building blocks of virtual images:

- Base image

The base image is the underlying virtual image that you use to create all of the other virtual images that you need. At a minimum, the base image contains an operating system.

- Software bundle

A software bundle is a container for one or more software products or components within a virtual image. Every virtual image that is produced by using ICCT has a base image and one or more software bundles.

Starting from a base image, you build a new image by adding one or more software bundles. Each software bundle includes a definition where you describe the included software and requirements and the installation and configuration processes and activation methods.

- Virtual image

A virtual image is the output of the ICCT. A virtual image can be used as a base image to build more virtual machines with the same content.

Building a virtual image for use with Workload Deployer-based technologies requires expertise from individuals in the following job roles:

- An operating system specialist to create the base operating system image
- A software specialist to create the software bundle or bundles
- An image builder to construct the virtual image

The full process of image build and creation is shown in Figure 4-15 on page 95 and described in the steps that follow the figure.



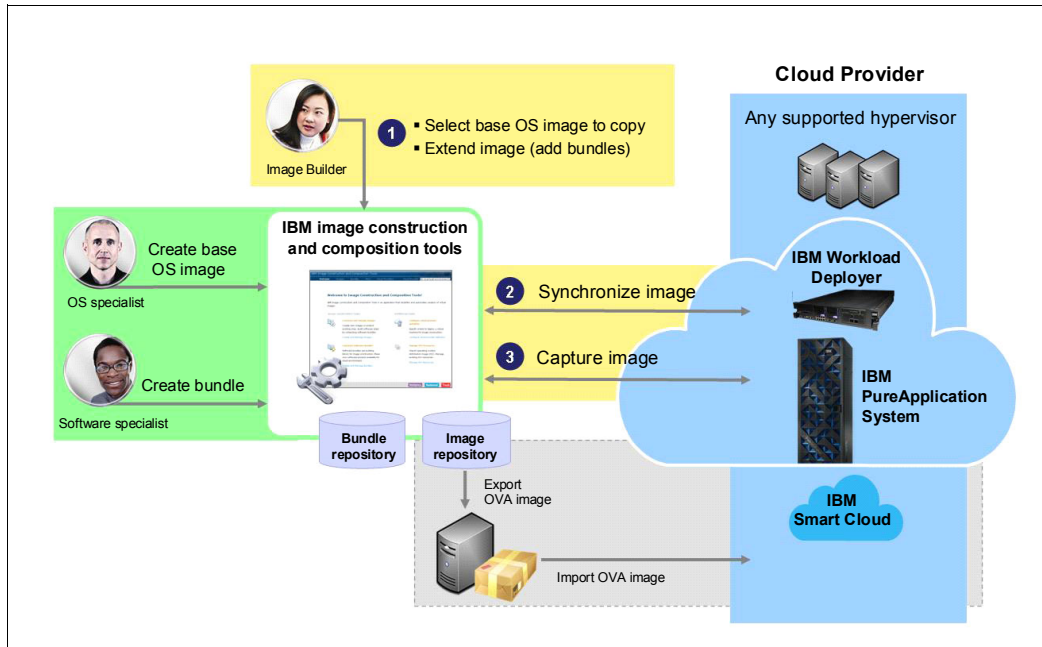


Figure 4-15 Process for building a custom image by using ICCT

Complete the following steps to build a virtual image with ICCT:

1. Select the base operating system image and extend it by adding the required software bundles.
2. Synchronize the image.

When you build an image on the ICCT console, a running instance of the image is not created until you run the synchronization. Complete the following steps:

- a. Deploy an instance of the base operating system image.
- b. Copy the software bundles that are required to build your own custom image and run the scripts that are associated with the bundles (these scripts typically install the software and complete some configuration steps).

When the synchronization completes, an instance of your custom image is running in your cloud environment.

3. Capture the running instance to make the custom image available in the ICCT catalog.

**Note:** For more information about using the IBM Image Construction and Composition Tool, see the following IBM Redbooks publications:

- *IBM Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud*, SG24-8011, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248011.html>

- *Creating Smart Virtual Appliances with IBM Image Construction and Composition Tool*, SG24-8042, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248042.html>

## Using a script package

A script package is an archive file that contains all of the artifacts that are needed to complete a customization action. The artifacts that are contained in a script package can be executable files, files that are needed by the executable files (such as .war and .ear files), or a JDBC driver for a specific database. Your options are almost unlimited.

A sample script package is shown in Figure 4-16.

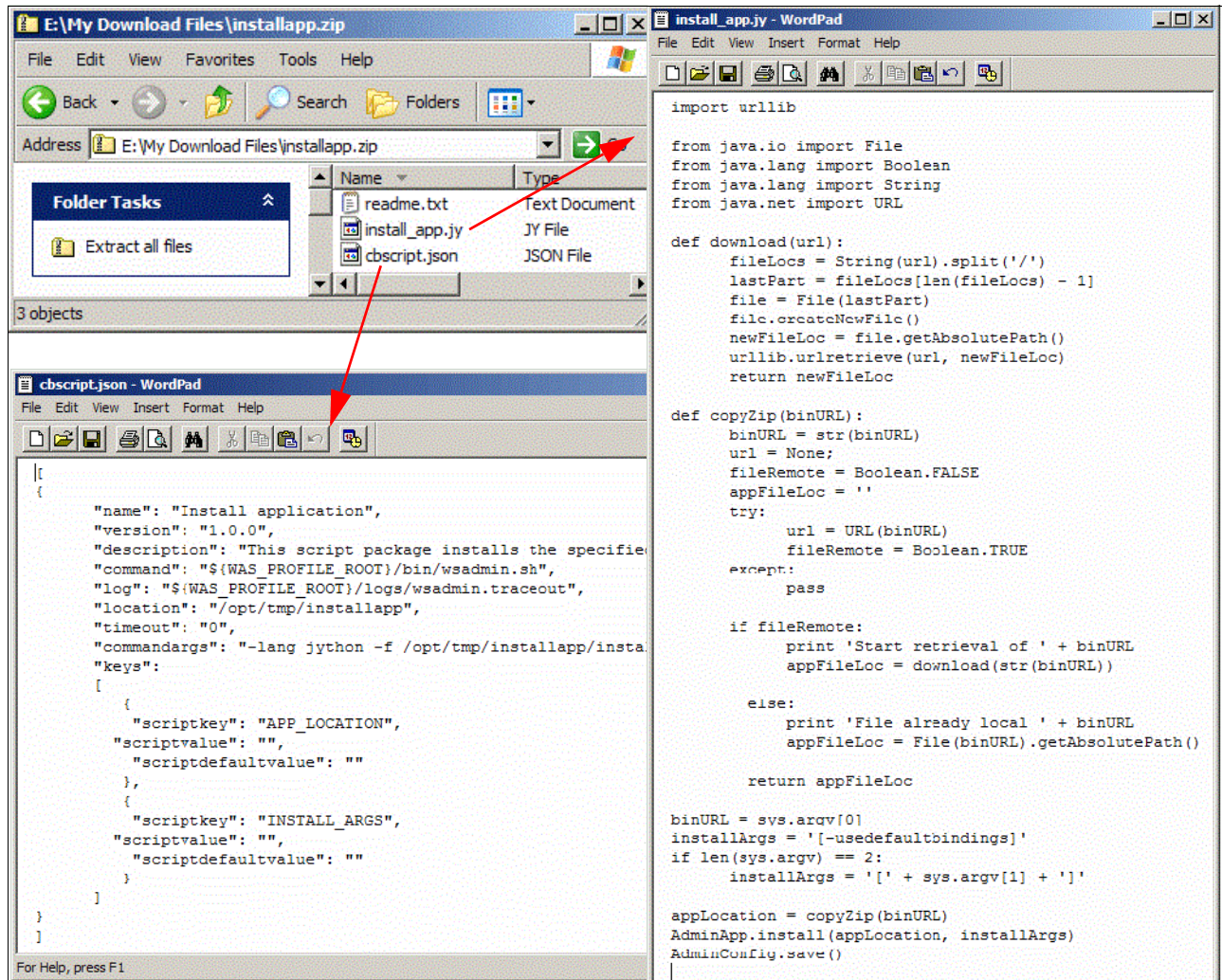


Figure 4-16 Sample script package

A script package can be used to customize an image in a way that is similar to extending an image. However, there is an important difference between these options. Unlike the extend and capture process, which is run once, a script package is run whenever a virtual image is deployed to compose a virtual system. This is disproportionately time-consuming when you consider the number of times the image must be deployed.

Given this limitation, it is important to understand and define what can be done with a script package versus what must be done through the extend and capture process.

A script package can be added to a virtual system pattern directly by using the pattern editor by dragging it to the part of the virtual system where script packages are run. Figure 4-17 shows an example of a virtual system pattern with script packages associated. This simple VSP is composed of a stand-alone WebSphere Application Server V8.5.0.2 part and an IBM DB2 V10.1.0.2 part. A script package that creates a database is associated with the DB2 part, while the script packages to install the DB2 drivers and to create and configure the required data source are associated with the WebSphere Application Server part.

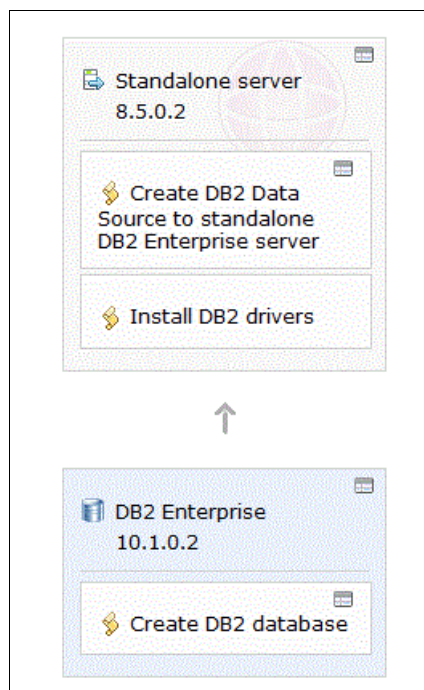


Figure 4-17 Sample virtual system pattern with script packages

Script packages can be run at the following times within the lifecycle of a virtual system instance. You can decide when to run the script package as you are creating it in the Workload Deployer catalog:

- ▶ At virtual system creation
- ▶ At virtual system deletion
- ▶ When it is initiated

A script package has a lifecycle of its own. Consider a script package that installs DB2 drivers. The package contains drivers for a specific version of DB2, so when you upgrade to a new version of DB2, you must also upgrade the script package. You upgrade the script package in the Workload Deployer catalog so that all future deployments of virtual systems that are using that script package automatically use the new version of DB2.

For more information about using script packages, see the IBM developerWorks article, *Customizing with WebSphere CloudBurst, Part 3: Using script packages for customizing above and beyond patterns*, which is available at this website:

[http://www.ibm.com/developerworks/websphere/techjournal/0911\\_stelzer/0911\\_stelzer.html](http://www.ibm.com/developerworks/websphere/techjournal/0911_stelzer/0911_stelzer.html)

## 4.2.2 Lifecycle of a deployed virtual system

In contrast to a virtual application in which all administration is done through the Workload Deployer console, you have the following options for managing a Virtual System Pattern:

- ▶ Manage the entire virtual system instance by accessing the images directly.
- ▶ Manage portions of the virtual system instance by using the tools of the cloud technology offering you are using.

In this section, we describe the second option and the following tasks that can be done by using the Workload Deployer console:

- ▶ Starting and stopping a virtual system instance
- ▶ Storing a virtual system instance
- ▶ Deleting a virtual system instance
- ▶ Applying a fix to a virtual system instance

### Starting and stopping a virtual system instance

The deployed virtual system instance can be started or stopped (as a whole) directly from your cloud environment console. Figure 4-18 shows the Start and Stop buttons in Workload Deployer (the Start button is shaded, which indicates that the sample system is already in a started state).

By clicking Stop, you stop the system as a single unit, so all of the virtual machines that make up the deployed system also are stopped.

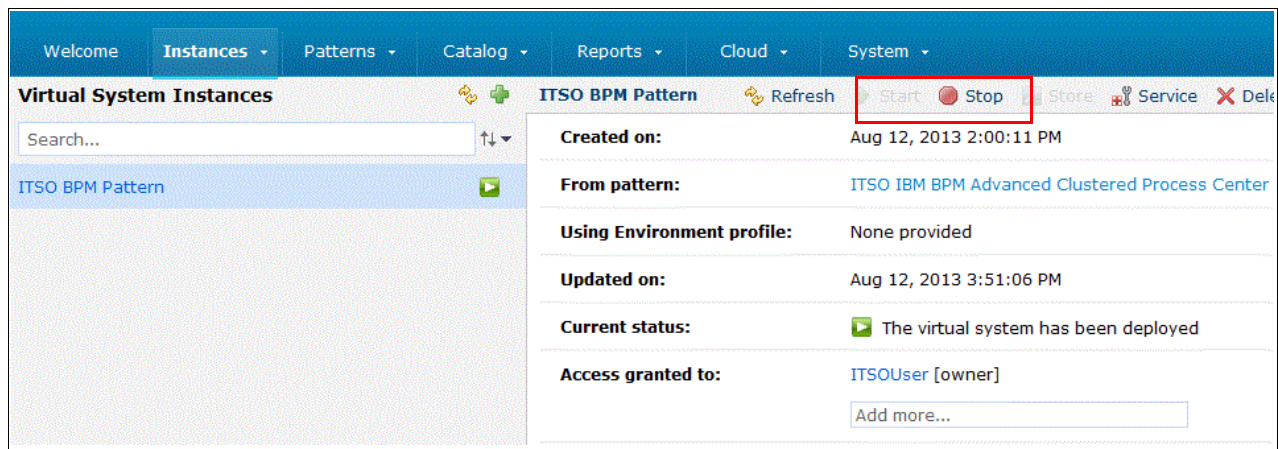


Figure 4-18 Starting and stopping a virtual system instance as a whole

Alternatively, you can manage the virtual images individually by using the buttons and other features that are available in the Virtual machines section of the system instance management window, as shown in Figure 4-19 on page 99.



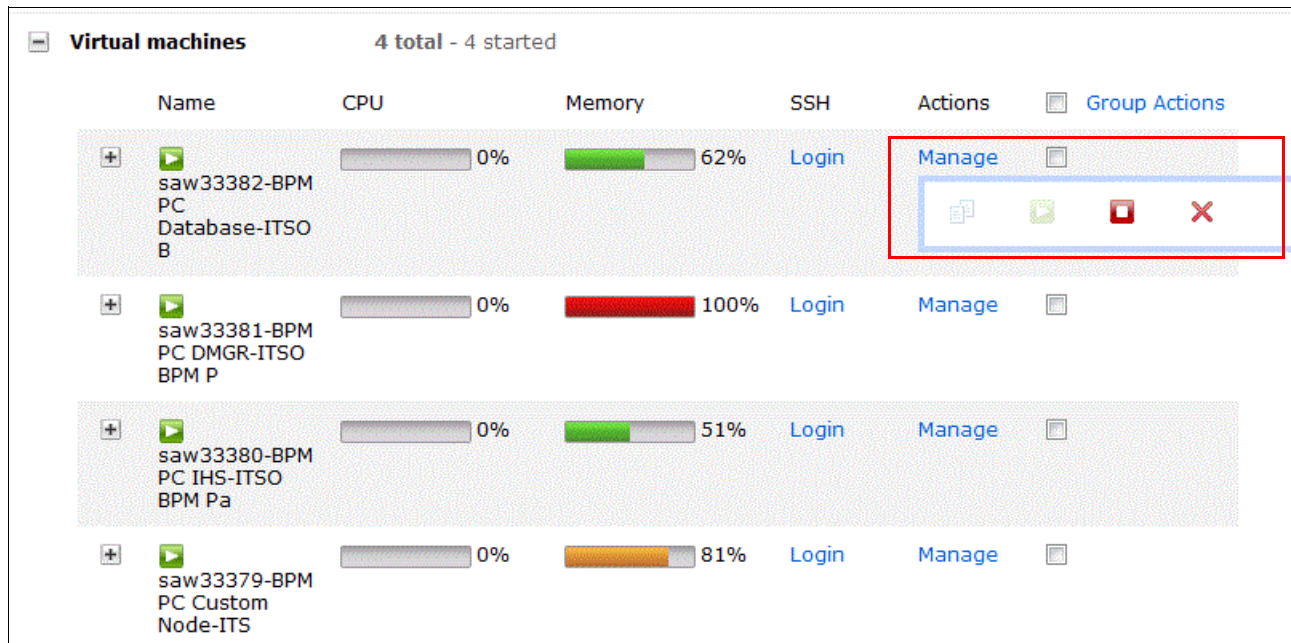


Figure 4-19 Managing virtual images individually

You identify the virtual machines that you want to start or stop under the Group Actions column by selecting **Manage**. You then submit the action by clicking **Group Actions** and selecting the action that you want to submit, as shown in Figure 4-20.

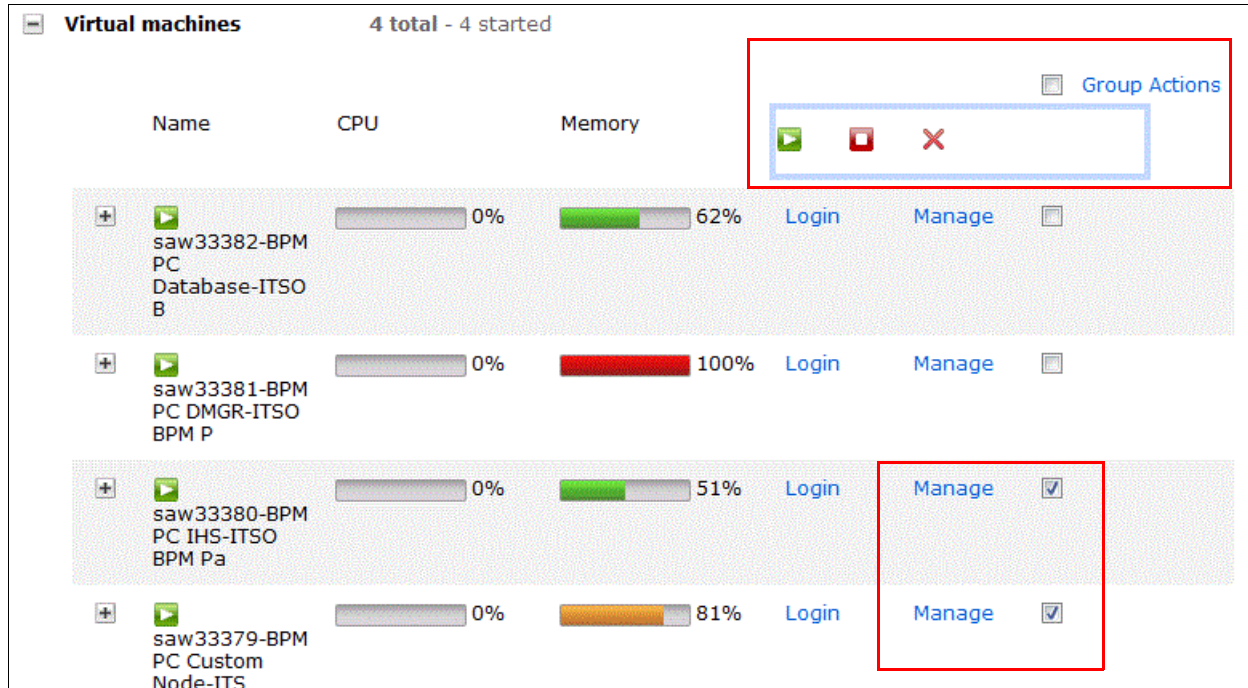


Figure 4-20 Managing virtual systems with Group actions

When you stop a virtual system instance, all of the resources are retained, which means that the physical resources (such as CPU and memory) and the system's software licenses and IP addresses are not returned to the pool and are not made available for other deployments. To release the resources, you must store the system, which is described next.

## Storing a virtual system instance

Storing is a good option if you do not intend to use the virtual system instance, but you do not want to delete it, either, perhaps because you hope to reuse it in the future.

Because a stored system is not deleted, not all of the associated resources are released when the system is stored. The store option releases all of the hypervisor resources, such as memory and CPUs, but it does not release the assigned IP addresses to avoid IP conflicts if you decide to restart the stored system in the future. The virtual system instance is still managed by the cloud environment.

To store a virtual system instance, first make sure that it is stopped. The instance must be stopped to be stored. Then, go to Virtual System Instances, select the virtual system instance you that want to store, and click **Store**. Figure 4-21 shows the Store button for the virtual system instance that is called ITS0 BPM Pattern.

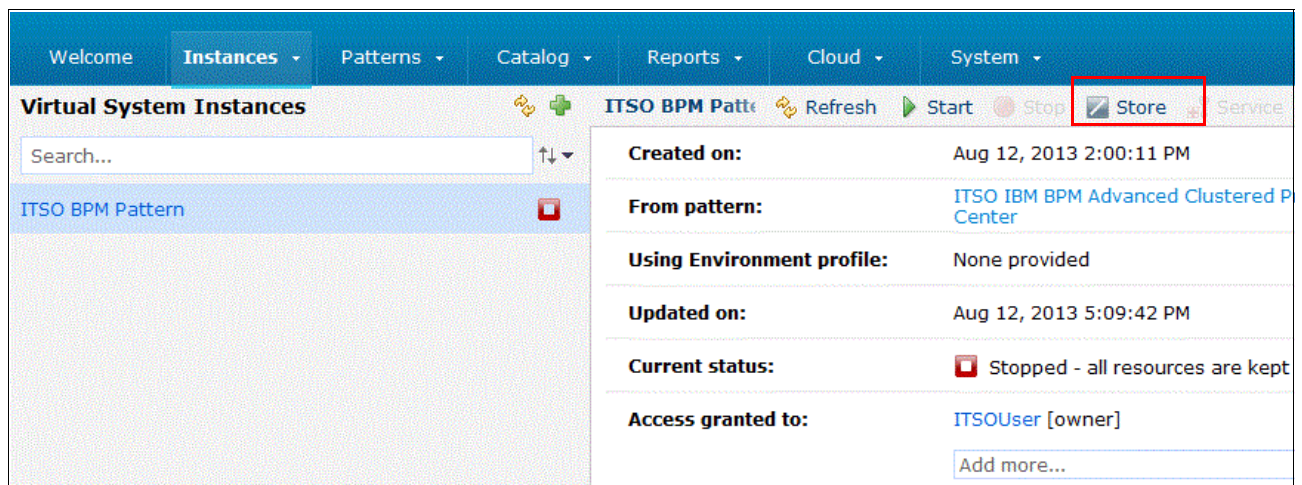


Figure 4-21 Storing a virtual system instance

## Deleting a virtual system instance

To delete a virtual system instance, click **Delete**, as shown in Figure 4-22.

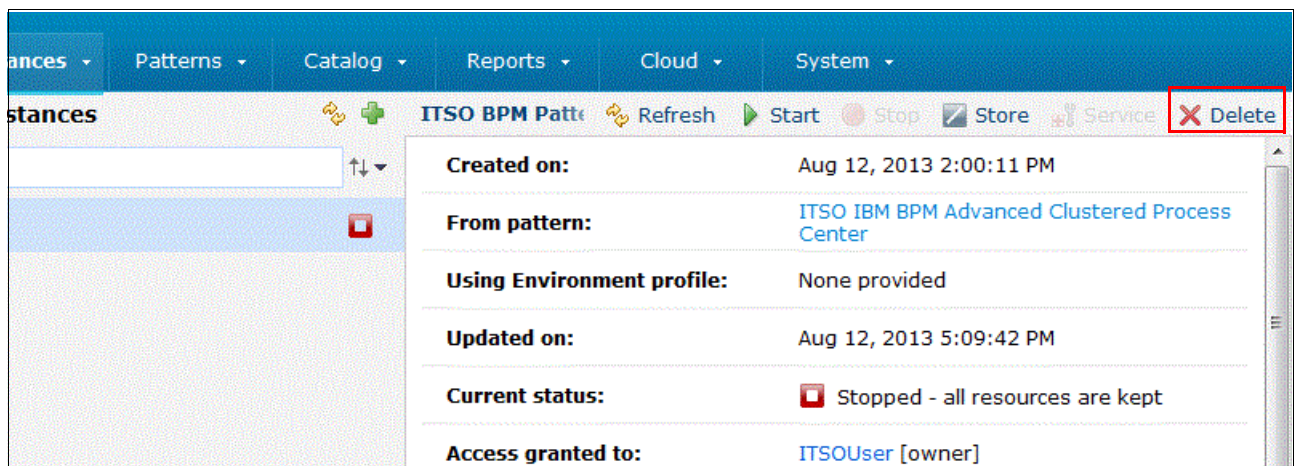


Figure 4-22 Deleting a virtual system instance

## Applying a fix to a virtual system instance

Sometimes you must change or adjust a running virtual system instance. These fixes can involve modifications to the operating system, middleware, or your image customizations.

### Fixing the operating system

Fixing the operating system can be done by using one of the following methods:

- ▶ Applying *emergency fixes*, which are similar to a script package.
- ▶ The use of the OS update service that is provided by the vendor.

Emergency fixes can be created by the user. This approach relies on a script package (a compressed file) that includes the following items:

- ▶ A `service.xml` file
- ▶ A custom shell script
- ▶ The artifacts that are needed for the update (for example, binaries)

The `service.xml` file is used to provide information that is required to apply the fix. Example 4-1 provides an example of this file. It includes information, such as what the emergency fix contains and the identity of the virtual image (the `ImagePrereqs` element in the example) that is needed for the fix to be applied. It also includes information about how to apply the fix, such as the command to be run, the working directory, and log file location (the `Package` element in the example).

Example 4-1 Sample `service.xml` file

---

```
<?xml version="1.0" encoding="UTF-8"?>
<rmsd:Service xmlns="<xmlschema_url>"
  xmlns:xsi="<xmlschema_instance_url>"
  xmlns:rmsd="<service_description"
  xsi:schemaLocation="<schema_location">
  <rmsd:ImagePrereqs>
    <rmsd:prereq name="<image_name>" version="<image_version>" />
  </rmsd:ImagePrereqs>
  <rmsd:Packages>
    <rmsd:Package name="<package_name>" type="ifix" target="APPLICATION">
      <rmsd:EnvVariables>
        <rmsd:EnvVariable
key="<var1_name>"defaultValue="<default1_value>"type="<var1_type>">
          </rmsd:EnvVariable>
        <rmsd:EnvVariable
key="<var2_name>"defaultValue="<default2_value>"type="<var2_type>">
          </rmsd:EnvVariable>
      </rmsd:EnvVariables>
      <rmsd:Command name="<script command>">
        <rmsd:Arguments>
          <rmsd:Argument><arg1></rmsd:Argument>
          <rmsd:Argument><arg2></rmsd:Argument>
        </rmsd:Arguments>
        <rmsd:Log><log_directory></rmsd:Log>
        <rmsd:Location><script_location></rmsd:Location>
        <rmsd:Timeout><timeout_value_ms></rmsd:Timeout>
      </rmsd:Command>
    </rmsd:Package>
  </rmsd:Packages>
</rmsd:Service>
```

---



For more information about the `service.xml` file, see the IBM Workload Deployer Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2Fwd%2Fpcr\\_servicexml.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2Fwd%2Fpcr_servicexml.html)

The `service.xml` file defines the command to be run. This is a custom shell script, so it can be used to apply fixes to the operating system or to any other customizations you made to the image.

Additionally, the compressed archive contains the artifacts that are needed for the update. The artifacts can be the fix, other script packages, or whatever you need to apply your fix.

To apply the fix, you first must upload it in your cloud environment catalog from the GUI for your environment. To upload the fix, click **Catalog** → **Emergency Fixes**, as shown in Figure 4-23.

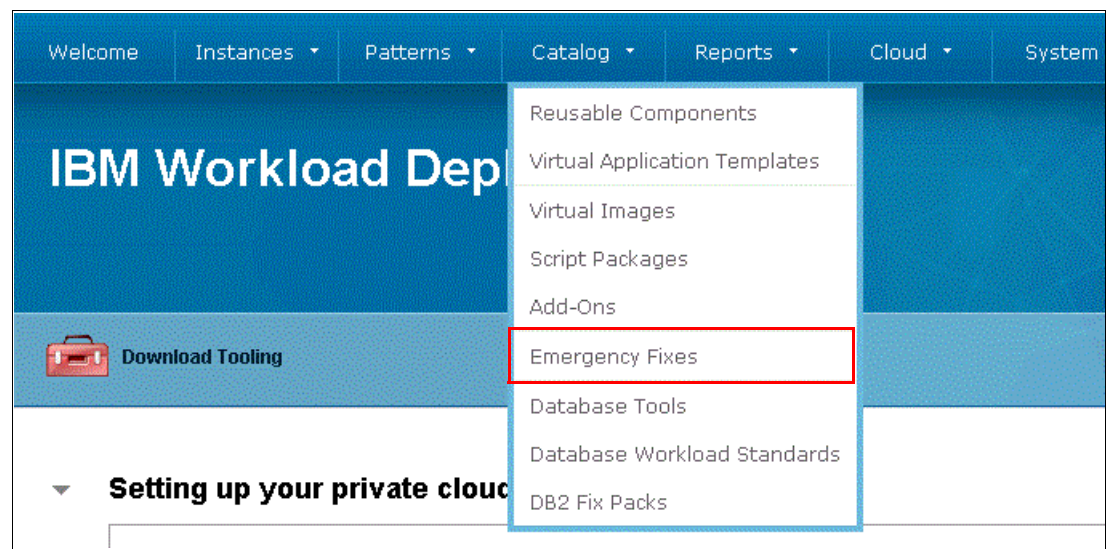


Figure 4-23 Accessing the Emergency fixes panel in the cloud environment catalog

After you upload your emergency fix, you can add images to the list of images to which the fix is applied, as shown in Figure 4-24.

Reports ▾ Cloud ▾ System ▾

OS Fix Refresh Delete

**Access granted to:** ITSUser [owner]  
Add more...

**Severity:** Normal ▾

**Applicable to:**

**Images:**

- IBM OS Image for Red Hat Linux Systems, RedHat Enterprise Linux 64-Bit (RHEL 6.2 X64) [remove]
- IBM Business Process Manager Advanced 8.0.1.0 RHEL 6 x64 (VMWare), RedHat Enterprise Linux 64-Bit (RedHat Enterprise Linux 6) [remove]
- IBM BPM Extended, RedHat Enterprise Linux 64-Bit (RedHat Enterprise Linux 6) [remove]

Add more...

**Plugins:** Add more...

Figure 4-24 Adding applicable images to the list for a fix

Images are one of the basic building blocks of virtual system patterns. You can apply an emergency fix to any running VSP instances that are based on the virtual image that must be fixed.

After the emergency fix is uploaded into the catalog and the applicable images are selected, you go to the virtual system instance management window for the VSP instance you want to fix and click **Service** to apply the fix. You can then choose some options to apply to your service request, such as the schedule or which fix you want to apply. Figure 4-25 on page 104 shows the emergency fix that is called OS Fix is selected.

Describe your service request.

☒ Schedule service

☒ Select service level or fixes

☐ Move to service level

☒ Apply emergency fixes

☒ OS Fix

☒ Product administrator user name and password

A snapshot will be taken before applying service.

OK Cancel

Figure 4-25 Service request details for the OS Fix emergency fix

For more information about creating your own emergency fix, see the IBM developerWorks article at this website:

[http://www.ibm.com/developerworks/websphere/techjournal/1001\\_amrhein/1001\\_amrhein.html](http://www.ibm.com/developerworks/websphere/techjournal/1001_amrhein/1001_amrhein.html)

More information also is available in the PureApplication System Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/topic/com.ibm.puresystems.appsys.1500.doc/iwd/pct\\_package\\_ef.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/topic/com.ibm.puresystems.appsys.1500.doc/iwd/pct_package_ef.html)

You can also use the appropriate update service from your OS vendor.

The Hypervisor Edition images that are provided by IBM are based on Red Hat Linux, so you can use the Red Hat OS Update Service to apply OS fixes for these images. This option is described in 4.4, “Red Hat OS Update Service” on page 120.

Hypervisor Edition images that are based on AIX and images that are based on Microsoft Windows can be upgraded by using IBM Endpoint Manager. For more information about the IBM Endpoint Service, see this website:

[http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.puresystems.appsys.1700.doc%2Fiwd%2Fapc\\_tem\\_relay.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.puresystems.appsys.1700.doc%2Fiwd%2Fapc_tem_relay.html)

### **Fixing the middleware**

The following options are available for fixing the middleware:

- Use emergency fixes

In addition to creating your own emergency fixes, you can use emergency fixes that are provided by IBM or by the provider of any third-party image you are using.

Fixes that are provided by IBM can be downloaded from Fix Central, which is available at this website:

<http://www-933.ibm.com/support/fixcentral/>

After you identify and download the fix that you want to apply, you must upload the fix to the catalog of your cloud environment. To do this, click **Catalog** → **Emergency Fixes** and then click **Add** to upload the fix. For security reasons, only .zip, .tgz, and .pak files can be uploaded.

- Wait for a new Hypervisor Edition image

IBM provides new Hypervisor Edition images whenever a new fix pack is released. These new images can be uploaded onto the system catalog after which you can redeploy your system by using one of the new images. To deploy the system and avoid the need to reconfigure the middleware, you can use an external tool, such as Rational Automation Framework or the Advanced Middleware Configuration (AMC) tool to capture the system configuration and reapply it to the updated virtual system instance.

PureApplication System offers the Advanced Middleware Configuration as a Hypervisor Edition image that provides the capabilities of Rational Automation Framework. You can create a VSP and deploy it on PureApplication System to onboard applications to the system or manage the configuration of the application that is already running on the system.

For more information about Rational Automation Framework or the Advanced Middleware Configuration tool, see the following resources:

- *Virtualization with IBM Workload Deployer: Designing and Deploying Virtual Systems*, SG24-7967, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg247967.html?Open>

- *Adopting IBM PureApplication System V1.0*, SG24-8113, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248113.html?Open>

- Use an external tool to apply the fix

Tools such as Rational Automation Framework can be used not only to manage the configuration of a running system, but to apply fixes to a running system.

- Apply the fix in the way that you typically use in your existing infrastructure

After a virtual system instance is deployed, you can administer it as you do a standard, non-virtual system. Applying upgrades can be done by using the tools that are included with each software product. For IBM products, these tools can include IBM Installation Manager or IBM Update Installer.

## 4.3 Virtual application lifecycle

The virtual application capabilities in IBM cloud technologies are based on the concept of standardized, application-centric pattern solutions. By using standard patterns, developers of applications in cloud environments can focus on the application and its requirements instead of the middleware infrastructure and the often complex configuration of the middleware products

VAPs are the deployment unit for a virtual application. VAPs define the resources that are required to support virtual applications, including web applications, databases, and user registries.

The underlying structure of VAPs is based on pattern types. Pattern types are the containers of solution-specific and topology-specific resources that are required for different types of virtual applications. Pattern types also provide shared services, including runtime infrastructure functions, such as caching services and elastic load balancing.

Pattern types contain plug-ins that include the parts of the application and provide lifecycle management of the parts (installation, configuration, start, stop, failure, and recovery, and so on). The plug-ins contribute the components, links, and policies that are used to assemble virtual applications. Before a virtual application is built by using a VAP, you must enable the pattern types that are needed to provide the components of the pattern.

A VAP consists of the following components, links, and policies:

- ▶ Components represent functional profiles for a workload, such as web applications and databases.
- ▶ Links define a connection between components in the pattern.
- ▶ By using policies, such as the JVM Policy, you can specify functional and non-functional requirements for your application environment.

When you create a VAP, you are creating a logical description of your virtual application. During deployment, this model is converted into a physical topology.

The foundation of the physical topology is the base operating system image, which you select in the default deployment settings of your cloud technology.

This section describes the following considerations regarding the individual lifecycles of the components that are part of the virtual application:

- ▶ Base operating system image lifecycle
- ▶ Deployed virtual application lifecycle:
  - Starting and stopping the virtual application instance
  - Managing the virtual application instance
  - Upgrading the virtual application instance
  - Maintaining the virtual application instance
  - Deleting the virtual application instance

### 4.3.1 Base operating system image lifecycle

IBM cloud technologies include a base operating system image that is used as a foundation for VAP components. This default image is stored in the virtual image catalog with the virtual images that are used for VSPs. However, unlike those other images, this image does not include any preinstalled middleware.

IBM provides the following base operating system image options:

- ▶ IBM OS Image for AIX Systems: This is an AIX image for use with PowerVM hypervisors. Different versions of the image are available that are based on disk size (60 GB, 105 GB, 135 GB, 465 GB, and 1551 GB).
- ▶ IBM OS Image for Red Hat Linux Systems: This is a Linux image for use with VMWare ESX hypervisors.



To select an image to use for all VAP deployments, go to the workload console and select **Cloud** → **Default Deploy Settings**, as shown in Figure 4-26.

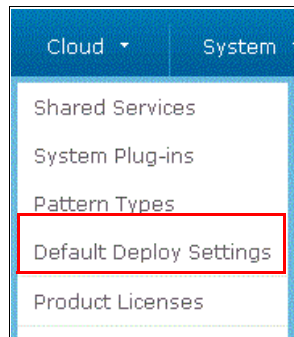


Figure 4-26 Accessing the default deployment settings

Depending on which IBM cloud technology is used, you can select the VMWare ESX image, the AIX Systems image, or both.

**Note:** In Workload Deployer, which supports VMWare ESX and PowerVM hypervisors, you can select the VMWare ESX image and the AIX Systems image. However, in PureApplication System, which is based on a specific hardware and hypervisor technology, you can select only the VMWare ESX image on W1500 or the AIX Systems image on W1700.

Figure 4-27 on page 108 shows the Default Deploy Settings panel in Workload Deployer. In this case, only one version of IBM OS image for Red Hat Linux Systems is available, but you can choose between different versions of the IBM OS image for AIX Systems.

IBM Workload Deployer

Welcome
Instances
Patterns
Catalog
Reports
Cloud
System

Default Deploy Settings

Define the default virtual image for deploying shared services and virtual applications

NOTE: Only supports 64-bit hypervisors and images.

Hypervisor Type: ESX

Set the default image below:

Name	License Agreement	Version	Description	Reference ID
IBM OS Image for Red Hat Linux Systems	Accepted	2.0.0.1	IBM OS Image for Red Hat Linux Systems	58

Change

Hypervisor Type: PowerVM

Set the image candidates in the list below:

Name	License Agreement	Version	Memory	Disk Size	Description
IBM OS Image for AIX Systems	Accepted	1.0.0.1	3072MB	31GB	IBM OS Image for AIX Systems
IBM OS Image for AIX Systems - Tiny	Accepted	1.1	3072MB	60GB	
IBM OS Image for AIX Systems - Small	Accepted	1.1	3072MB	105GB	

Figure 4-27 Default deployment settings for IBM Workload Deployer

To choose your wanted image, start by clicking **Change** within a particular hypervisor section. Select the image that you want to set as the default and then click **Update**, as shown in Figure 4-28 on page 109.

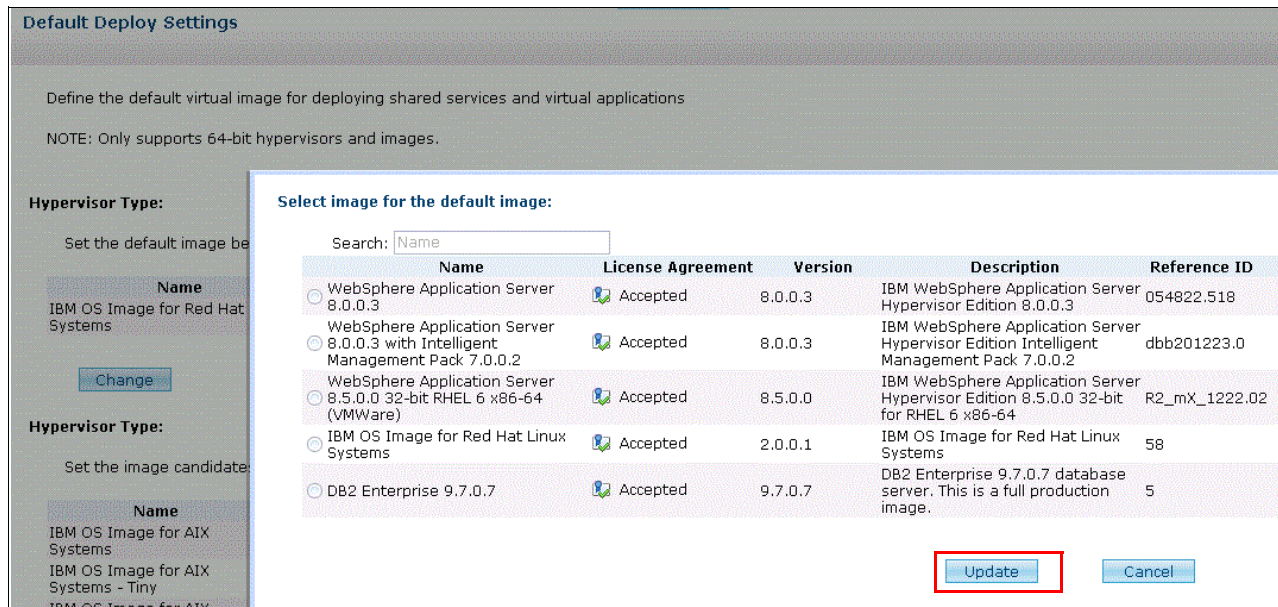


Figure 4-28 Selecting default base OS image for VMWare ESX hypervisor deployment

After you change the base image, all new deployments use the image that you defined.

### 4.3.2 Deployed virtual application lifecycle

As with the virtual systems that we described earlier, virtual application instances also have a lifecycle. This section describes the key lifecycle activities that can be done directly from the cloud environment.

#### Starting and stopping a virtual application instance

After you deploy a virtual application, you can start and stop it directly from the workload console of the cloud environment. To display the available management options, click **Instances** → **Virtual Application Instances** and then select the instance with which you want to work. Figure 4-29 shows the management console of a sample web application instance that is used here to describe the different actions that can be performed.

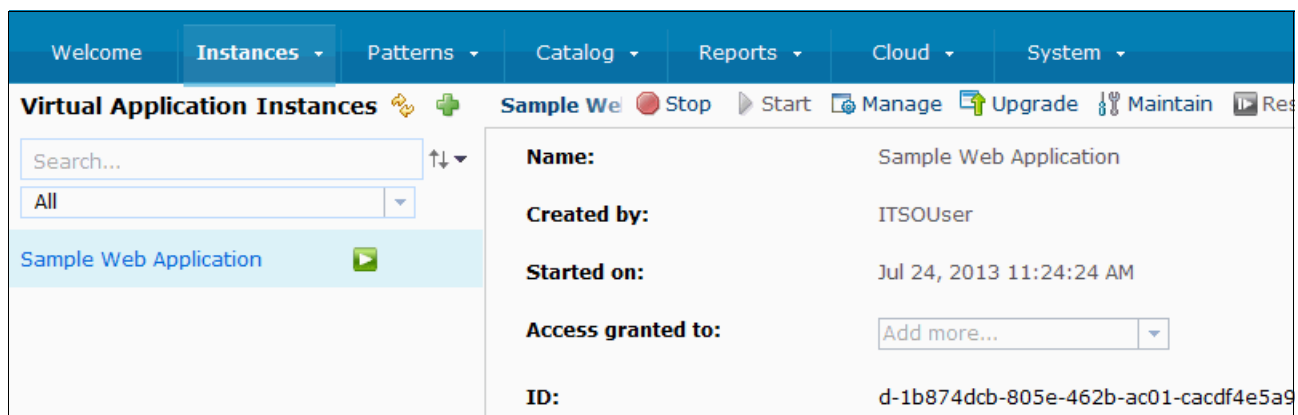


Figure 4-29 Management console for the sample web application

The sample application that is shown in the figure is deployed by using Web Application Pattern Type 2.0.0.1, but the concepts that are explained here apply to any virtual application pattern.

After you select the deployed virtual application, you can start it or stop it by clicking the appropriate button, as shown in Figure 4-30.

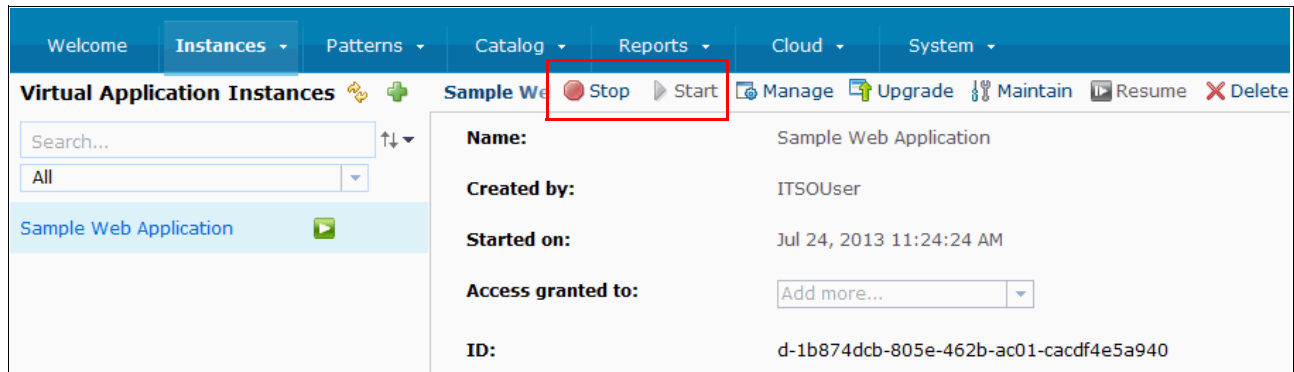


Figure 4-30 Starting and stopping a virtual application

## Managing a virtual application instance

To manage the virtual application, go to the same Default Deploy Settings panel and click **Manage**, as shown in Figure 4-31. The Virtual Application Console opens, as shown in Figure 4-32 on page 111.

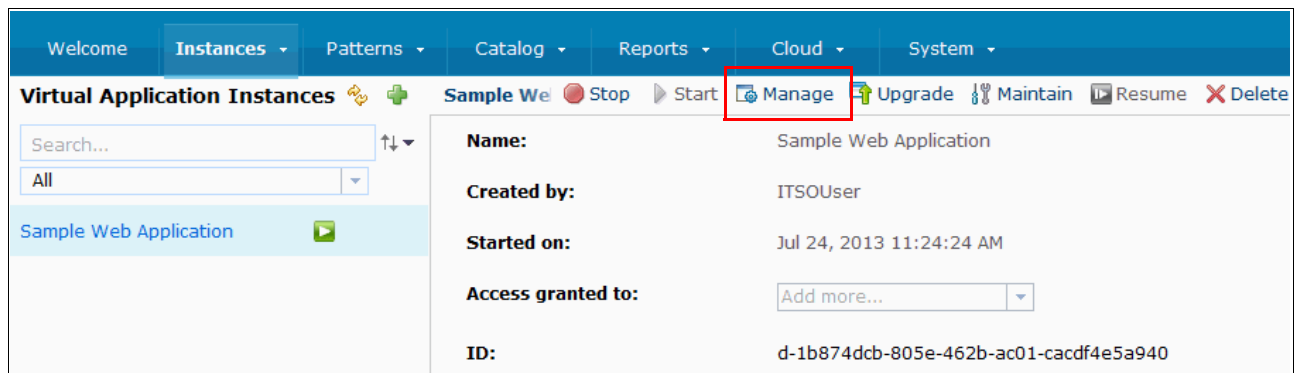


Figure 4-31 Initiating management of a virtual application

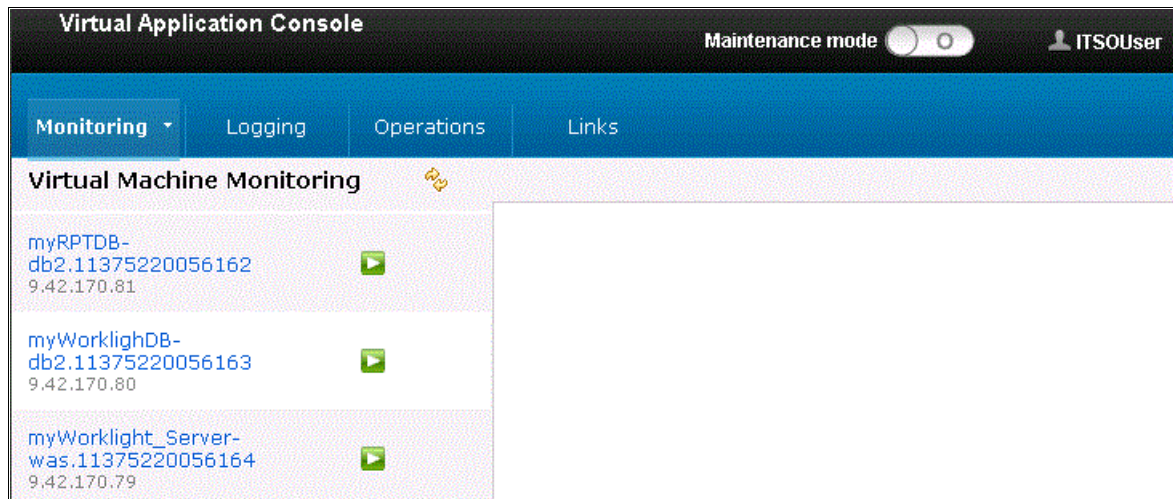


Figure 4-32 The Virtual Application Console

Through the Virtual Application Console, you can access the following panels:

- ▶ Monitoring
- ▶ Logging
- ▶ Operations
- ▶ Links

### Monitoring

The Virtual Machine Monitoring panel is displayed in the console by default and is used to monitor the deployed virtual machines and middleware. If this view is not already open, you can access it by selecting **Monitoring** → **Virtual Machine**.

Select the virtual machine that you want to monitor and the statistics about its memory, network, processor, and storage resources are displayed as diagrams on the right side of the window. The most recent data is always displayed. By hovering your mouse over the nodes in a particular diagram, you can see detailed data about that node.

Figure 4-33 shows sample diagrams for network and storage resources.

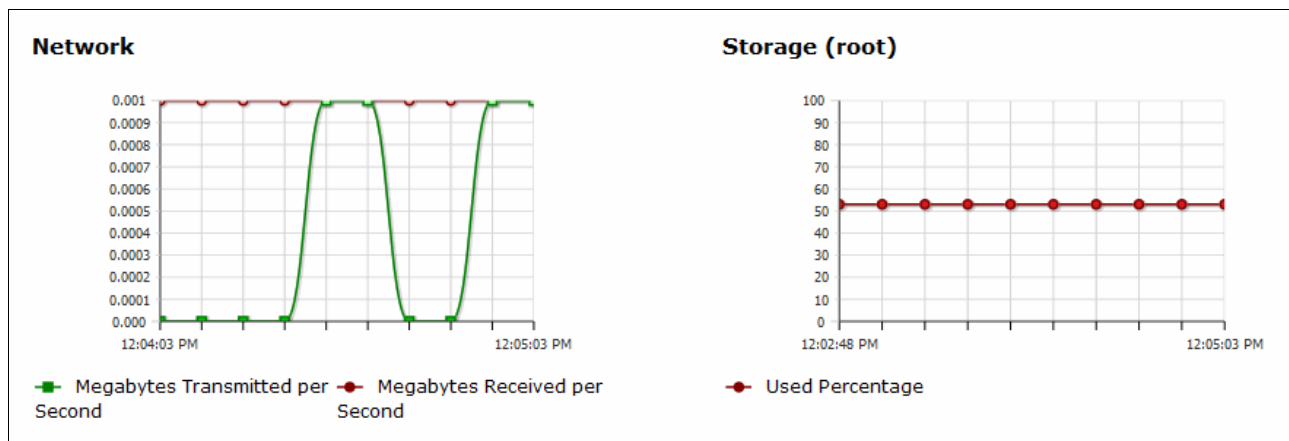


Figure 4-33 Diagrams in the Virtual Machine Monitoring view

The Middleware Monitoring view is used to monitor the status and performance of the middleware layer. To open this view, choose **Monitoring** → **Middleware**.



The middleware instances for the virtual application that is monitored are listed on the left side of the window, as shown in Figure 4-34. Click the middleware instance that you want to monitor to see its statistics that are displayed as diagrams. In the example that is shown in Figure 4-34, a single WebSphere Application Server instance is hosting the sample application.

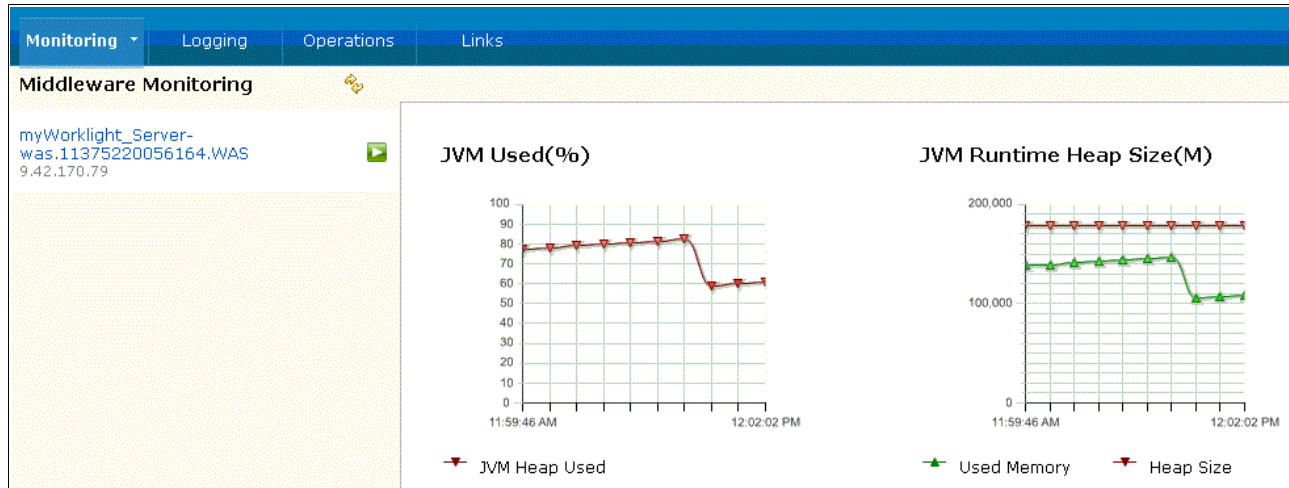


Figure 4-34 Diagrams in the Middleware Monitoring view

## Logging

You use the Logging panel to evaluate overall system health and for troubleshooting.

Start by downloading and viewing log files in the Log Viewer portion of the Virtual Application Console. Complete the following steps:

1. Select the Logging tab. This displays the available logs in a tree format on the left side of the window, as shown in Figure 4-35. In the example that is shown in Figure 4-35, there is only one virtual machine for the virtual application.

The logs are organized in categories for each virtual machine: OS (operating system), Middleware (database, application server, and so on), and IBM Workload Deployer Agent.

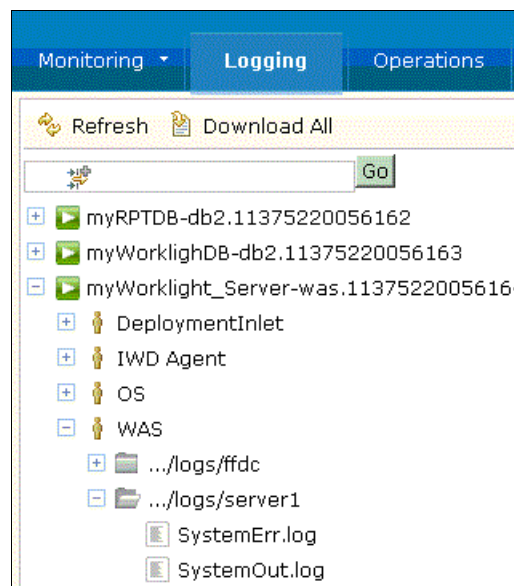


Figure 4-35 List of available logs on Logging tab

- Expand the navigation tree to see the full list of log files and then click the listing for the log file that you want to examine. The log is retrieved from the virtual machine and displayed on the right side of the window, as shown in Figure 4-36. Click **Monitor end of log file** to view the logs as they are written by the middleware.

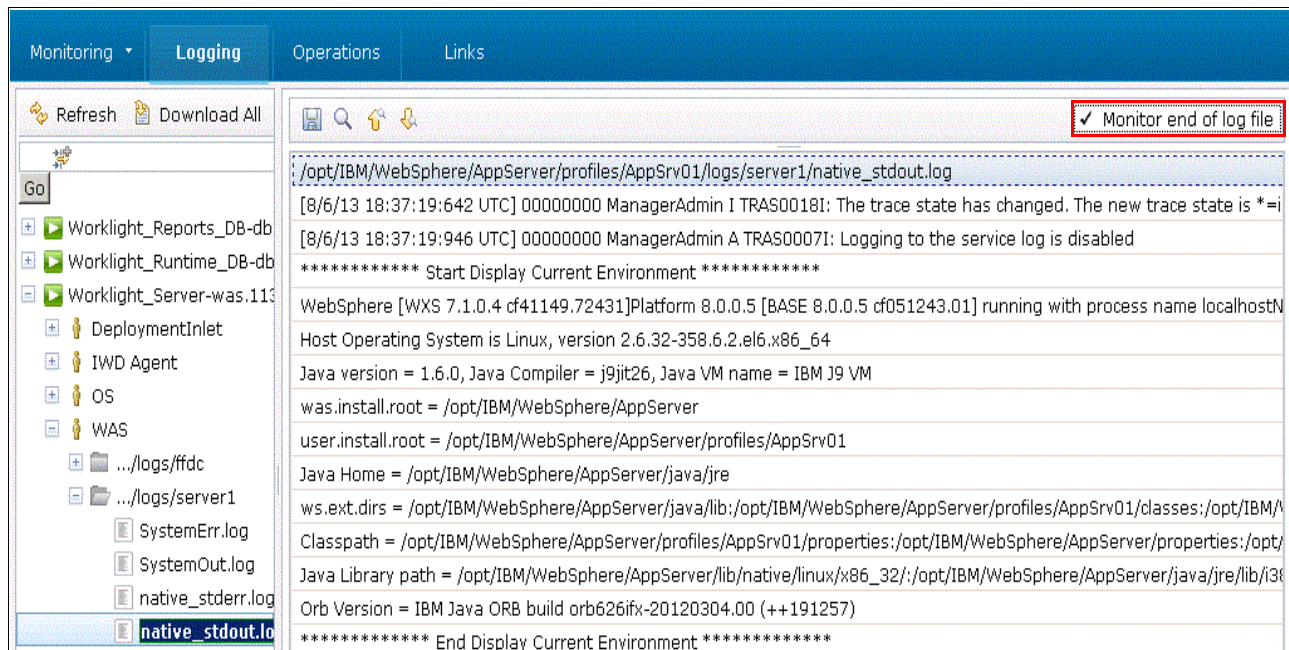


Figure 4-36 Enabling the function to monitor logs as they are written

- If you must have all of the log files available locally on your machine, you can download all of the logs by clicking **Download All**, as shown in Figure 4-37.

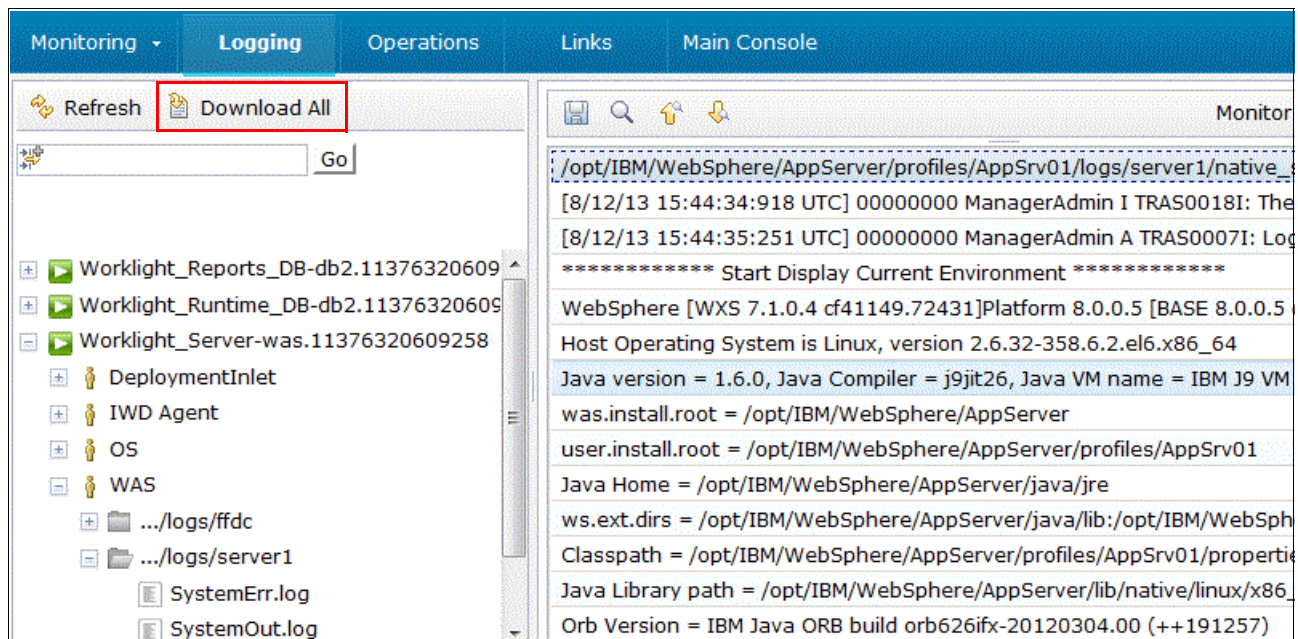


Figure 4-37 Download All option to save all log files to your local system

## Operations

You use the Operations panel to perform the following administrative tasks:

- ▶ Setting a different trace level

You can change the trace level to each of the instances that make up your pattern. For example, you can move to the WebSphere Application Server component of a virtual application and change the trace level while the application is running, as shown in Figure 4-38.

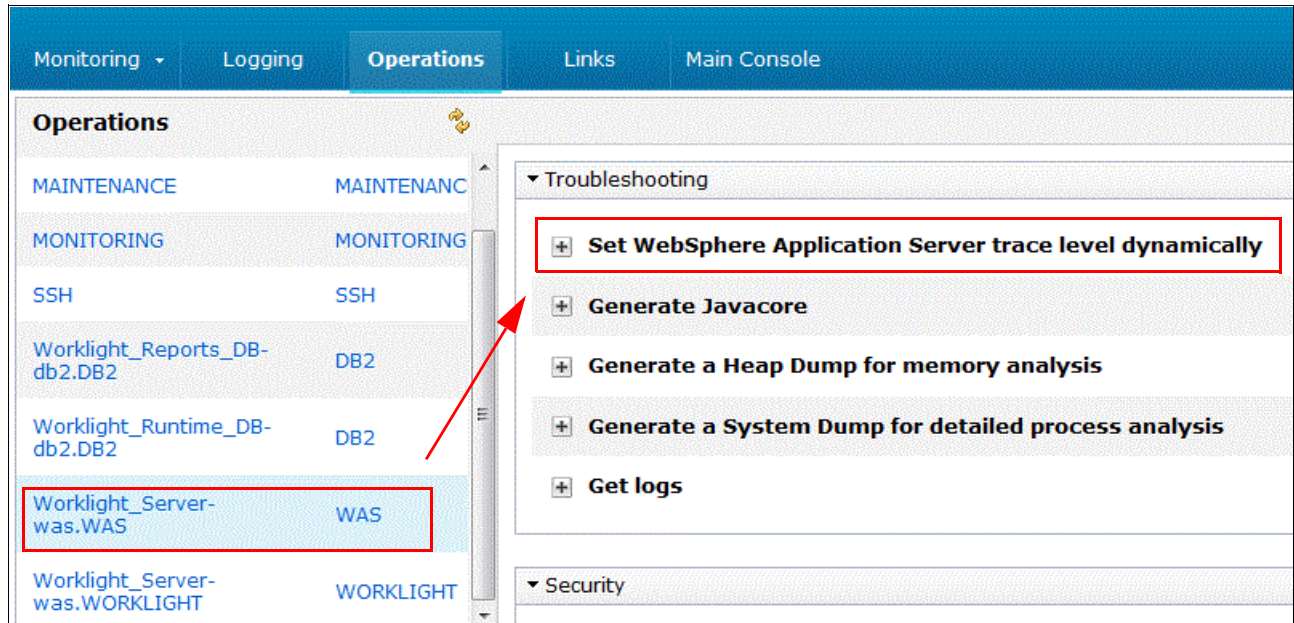


Figure 4-38 Setting the trace level for a running virtual application

- ▶ Applying a fix

As with virtual system instances, virtual application instances are composed of middleware software that might need to be fixed.

Before you can apply the fix, you must put the instance into maintenance mode by using the option at the top of the Virtual Application Console. After the virtual application is in maintenance mode, complete the following steps:

- Go to the Operations panel.
- Select the middleware instance that you want to fix.
- Expand the Update configuration section.
- Select the fix and submit the action.

After the update is complete, you can exit maintenance mode to bring the application back to an online status.

- ▶ Updating the middleware configuration

To change the middleware configuration, click **Operations** → **Update configuration** for the middleware instance that you want to update. Numerous configuration parameters can be changed.

The following examples of possible changes are related to WebSphere Application Server:

- Async response timeout
- Total transaction lifetime timeout
- Client inactivity timeout
- Maximum transaction timeout



- Maximum Session Count
- Enable Verbose Garbage Collection
- Class Loader Order

► Managing the application

By using the Operations view, you also can update the application. In the Update Configuration section, you can provide the appropriate .ear, .war, or .eba file to update your application.

## Upgrading a virtual application instance

A virtual application relies on pattern types. Pattern types are the containers of solution-specific and topology-specific resources that are required for different types of virtual applications. Virtual application pattern vendors can provide an updated version of a pattern. After the updated pattern is provided and you have virtual application instances running that are based on the previous instance of the pattern type, you must upgrade them.

Before you can upgrade a virtual application instance, you must import the new version of the pattern type into the cloud environment. You also must enable the pattern so it can be used.

To enable the new pattern type version, complete the following steps:

1. From the Workload Console, go to **Cloud** → **Pattern Types**, as shown in Figure 4-39. This takes you to the Pattern Types window, where all of the pattern types that are available on the cloud environment are displayed.

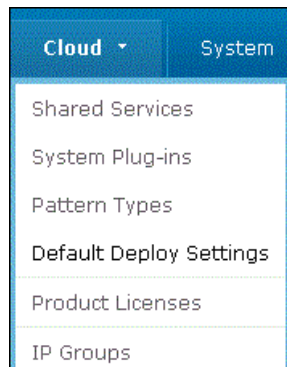


Figure 4-39 Accessing the Pattern Types window

2. From the pattern type list, select the pattern type that you want to enable. Figure 4-40 on page 116 shows the selection of Web Application Pattern Type 2.0, for which the following versions are available on the system (as indicated on the left side of the window in the figure):
  - Version 2.0.0.1, which has a green check mark next it to indicate that it is enabled and available.
  - Version 2.0.0.5, which has a yellow warning symbol beside it to indicate that it is unavailable and must be enabled before it can be used.
3. Click the listing for **Web Application Pattern Type 2.0.0.5**. A panel opens in which you must accept the license for the pattern type to use it.
4. Enable the new pattern type version by clicking **Enable**, as shown in Figure 4-40 on page 116.

With the new pattern type enabled, all new systems that are deployed and that use Web Application Pattern Type 2.0 use the new version, 2.0.0.5 from now on.

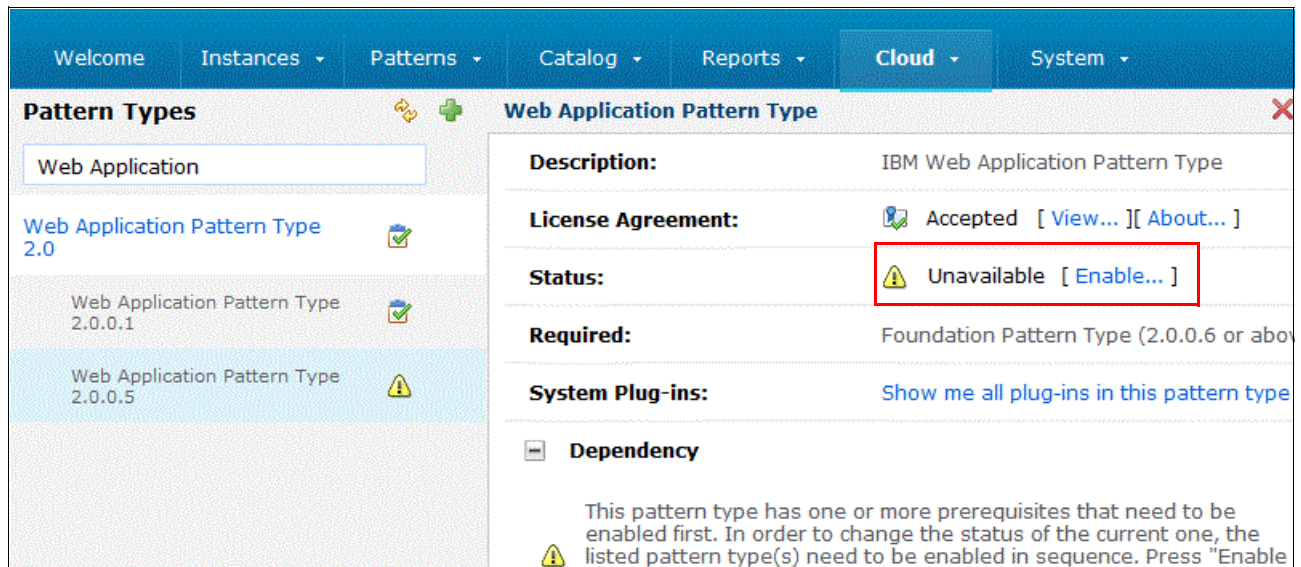


Figure 4-40 Enabling a new version of a pattern type

You can also upgrade any virtual application instances that were deployed by using the previous pattern type version to use the new one instead. To upgrade a running virtual application to use a new pattern type version, complete the following steps:

1. From the Workload console, go to the Virtual Application Instances page by clicking **Instances** → **Virtual Applications**. Select the running instance that you want to upgrade.
2. Click **Upgrade**, as shown in Figure 4-41.

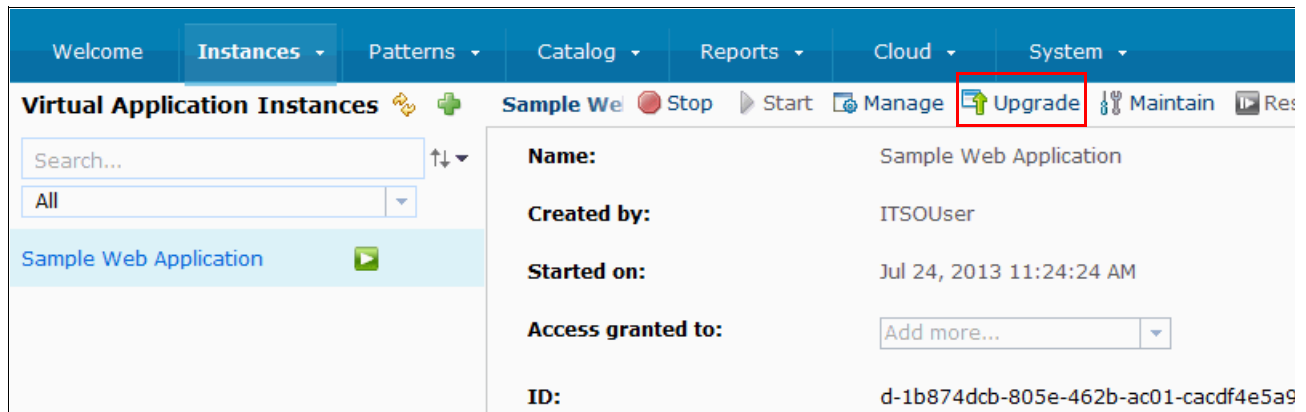


Figure 4-41 Upgrading a running virtual application instance, such as to use a new pattern type version

3. A window provides a summary of the components within the new pattern type version that is upgraded in the running instance. To proceed, click **Update**, as shown in Figure 4-42.

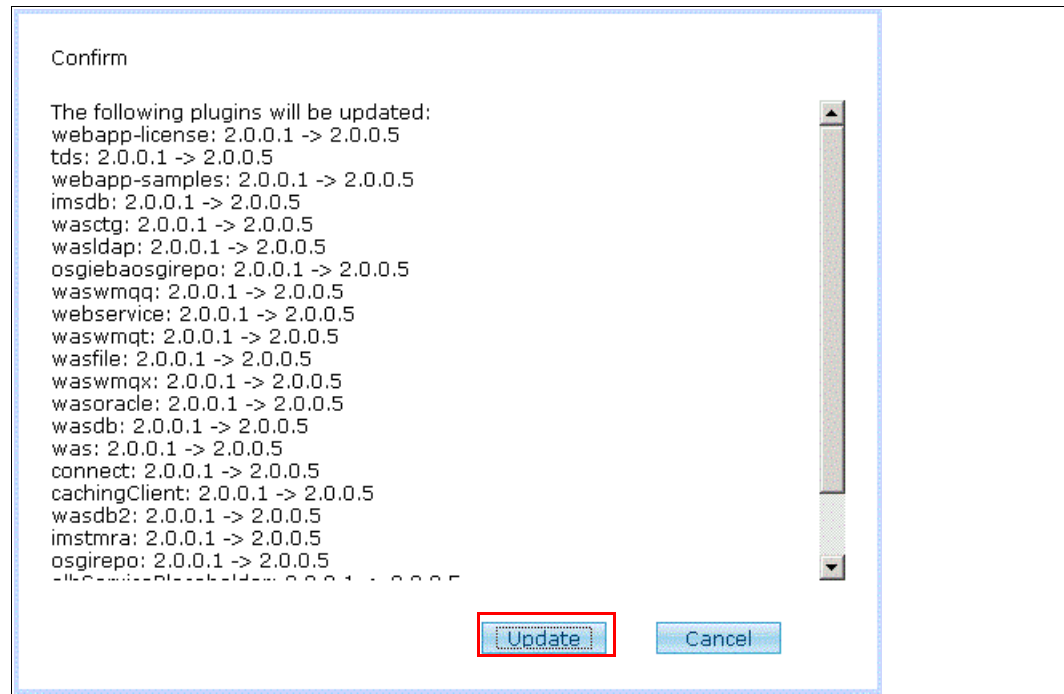


Figure 4-42 Confirming a decision to upgrade a running virtual application instance

You see a message that confirms that the upgrade process began, as shown in Figure 4-43.

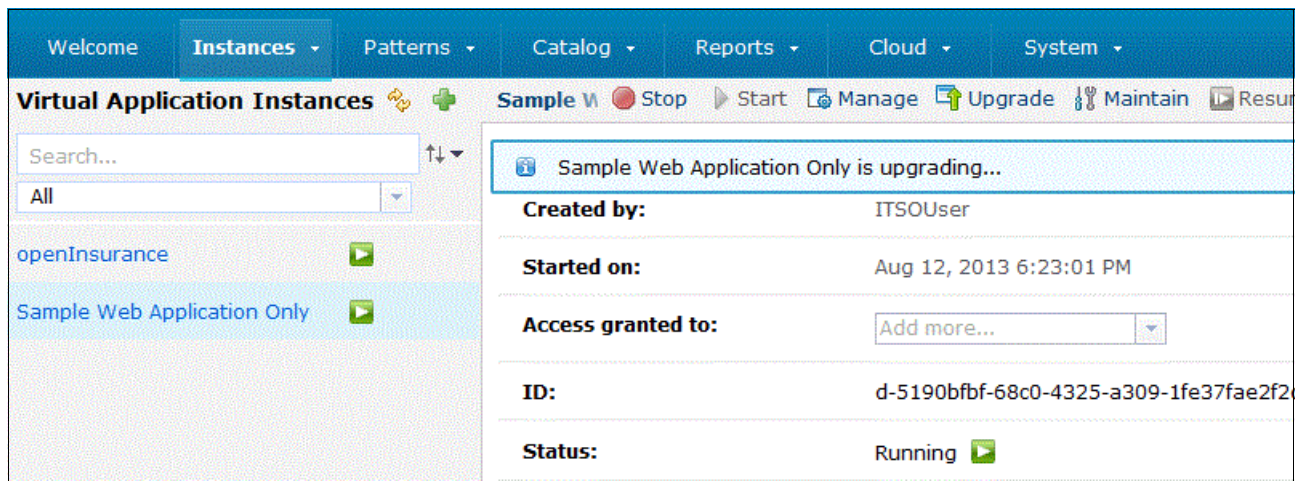


Figure 4-43 Message stating that a running virtual application instance is being upgraded

The application is then moved into maintenance mode, as shown in Figure 4-44.

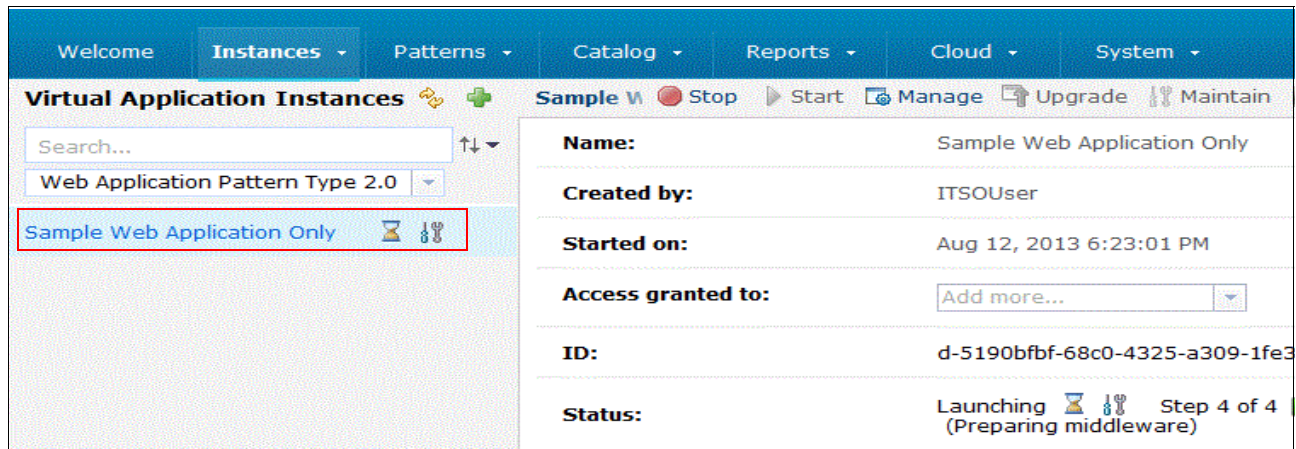


Figure 4-44 Application is moved to maintenance mode

After the virtual application instance upgrade is completed, it returns in a started state.

### Maintaining a virtual application instance

The pattern upgrade process upgrades the middleware that supports your application. Actual application maintenance follows a different process that is performed within a virtual application instance by using the Workload console.

Complete the following steps:

1. From the Workload console, go to **Instances** → **Virtual Applications** and select the virtual application instance that corresponds to the application you want to upgrade.
2. Click **Manage**, as shown in Figure 4-45.

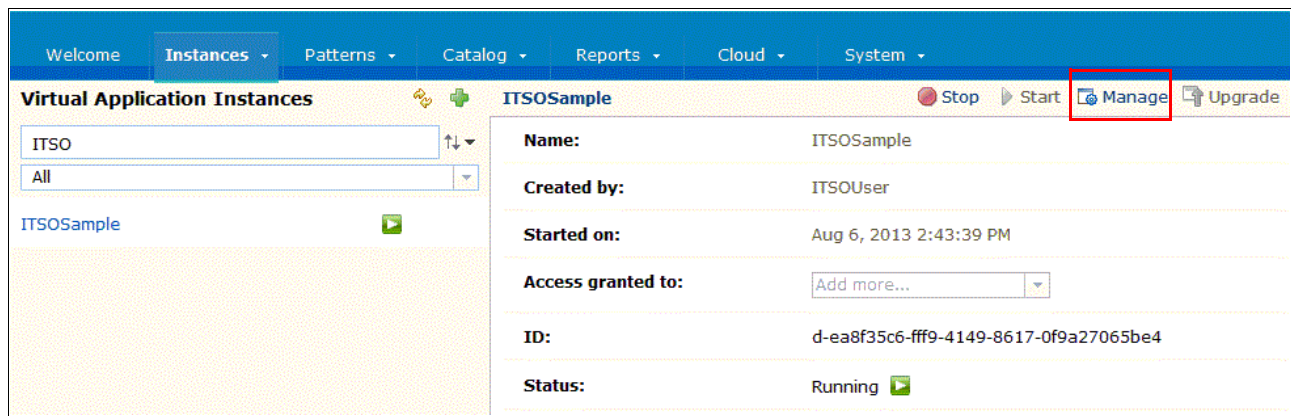


Figure 4-45 Managing a virtual application



3. Select the Operations tab, choose the application that you want to upgrade, and then expand the Update configuration field, as shown in Figure 4-46.

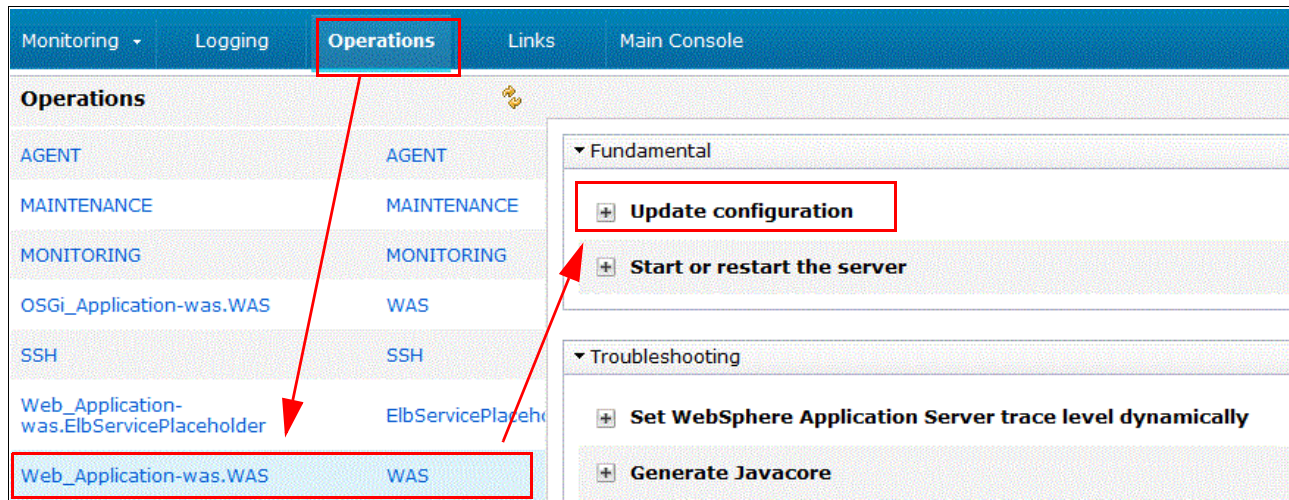


Figure 4-46 Selecting an application to upgrade and expanding the update configuration field

4. Click **Edit** to provide a new application version file, as shown in Figure 4-47.

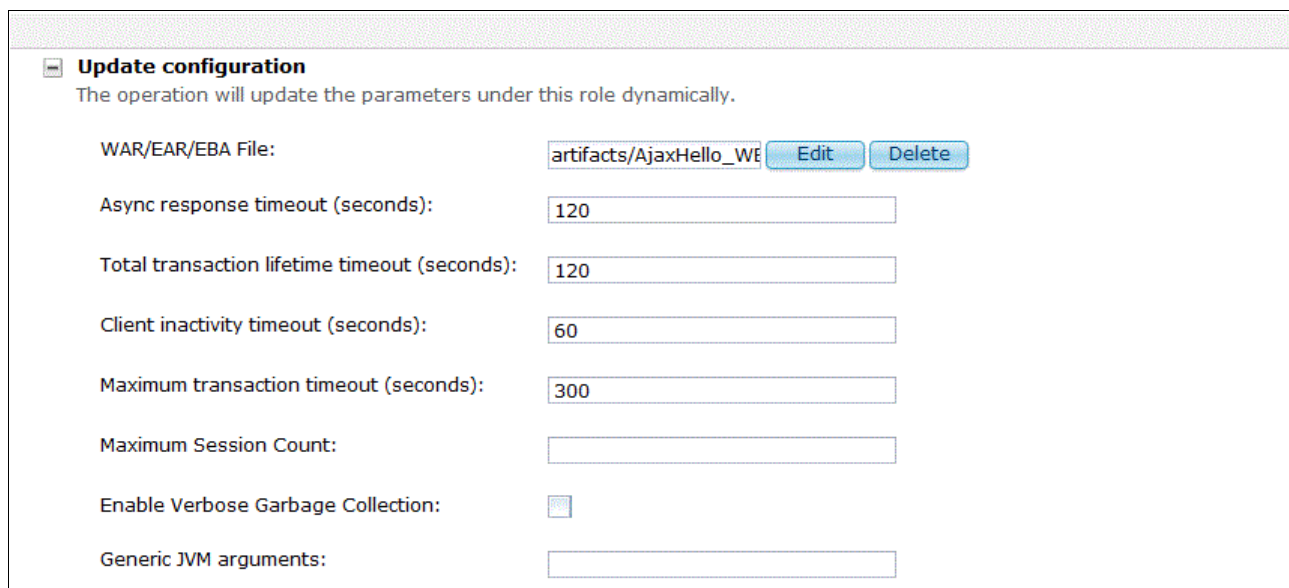


Figure 4-47 Updating the configuration options

5. Select the new application version by choosing a file from your local file system or by providing a URL where the file can be found, then click **OK**, as shown in Figure 4-48 on page 120. The application is automatically updated on the running virtual application instance.

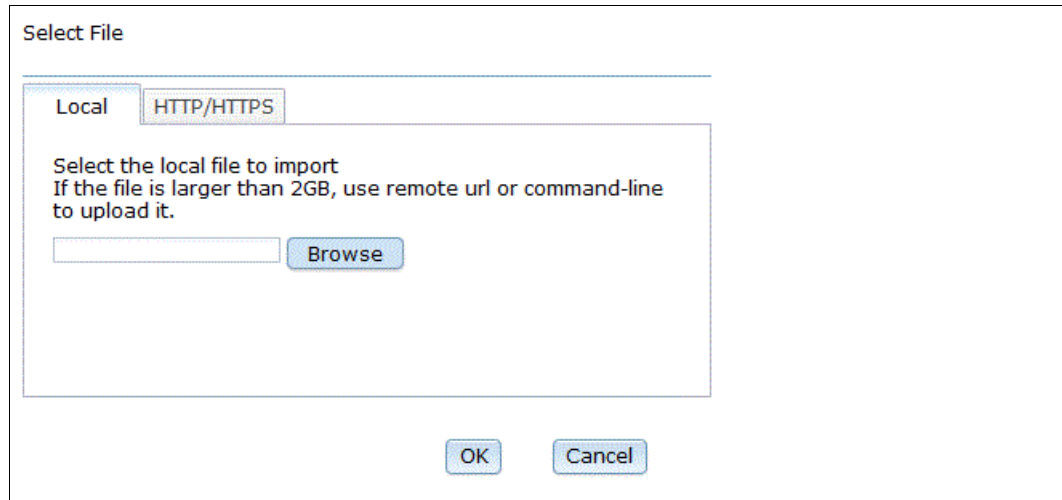


Figure 4-48 Selecting the new application file

### Deleting a virtual application instance

Complete the following steps to delete a running virtual application instance:

1. On the Workload Console, go to **Instances** → **Virtual Applications** and then select the virtual application instance that you want to delete from the list on the left side of the window.
2. On the virtual application instance management console, click **Delete**.

## 4.4 Red Hat OS Update Service

In “Fixing the operating system” on page 101, we described the two options that are available to fix the operating system of a virtual system or virtual application: applying so-called emergency fixes, or by using vendor-provided OS update services. It follows that to update a Red Hat based Hypervisor Edition, you can use the Red Hat OS Update Service, which is provided as a shared service for your cloud environment.

You can deploy this shared service so that images that are running Red Hat Enterprise Linux operating systems in virtual application and virtual system instances can receive updates and fixes directly from Red Hat as they become available. The shared service is based on the Red Hat Update Infrastructure (RHUI) model and can provide updates only to images that are running in the same cloud group as the service.

The Red Hat OS Update Service provides a Yellowdog Updater, Modified (YUM) repository. Clients with Red Hat Enterprise Linux operating systems that are in the same cloud group can access the YUM repository to download and install Red Hat Package Manager (RPM) updates and fixes directly from Red Hat.

Deploying the shared service requires administrative privileges. Complete the following steps:

1. Go to **Cloud** → **Shared Services** and select **Red Hat OS Shared Service** from the list on the left side of the window.
2. Click **Deploy**, as shown in Figure 4-49.

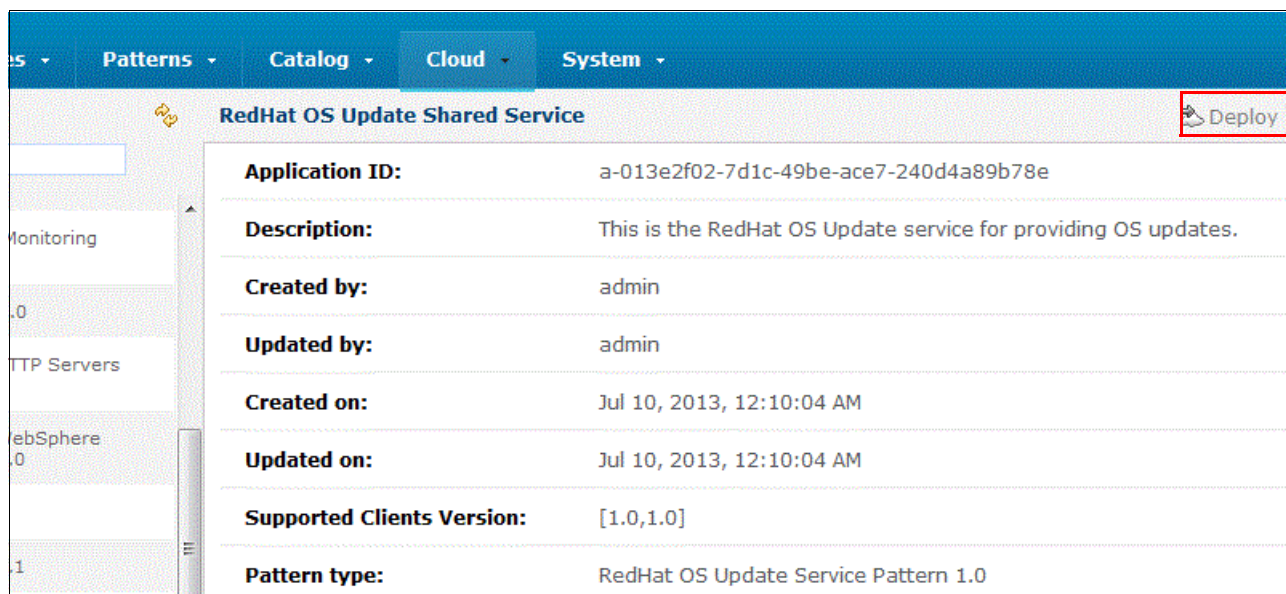


Figure 4-49 Deploying the Red Hat OS Update Service

When you deploy the Red Hat OS Update Service, three virtual machine instances are created, including two content distribution server (CDS) instances and one Red Hat Update Appliance (RHUA) instance. The CDS instance hosts the YUM repository for the cloud group. You must specify the storage size for the YUM repository when you are configuring the shared service. Optionally, you can provide information about open ports that allow your YUM repository to communicate directly with the external Red Hat network.

Figure 4-50 on page 122 shows an architectural overview of a deployed Red Hat OS Update Service.

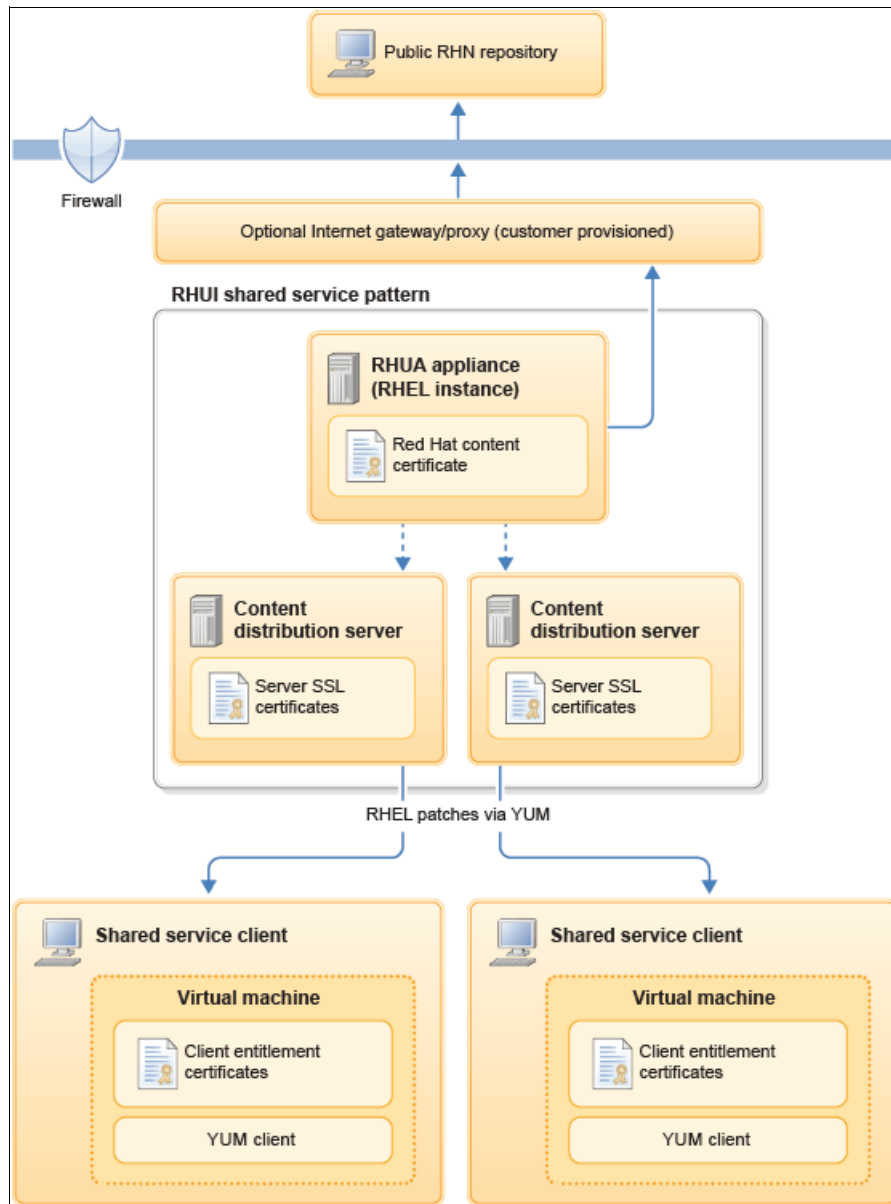


Figure 4-50 Architectural overview of a deployed Red Hat OS Update Service

## 4.5 Monitoring deployed instances

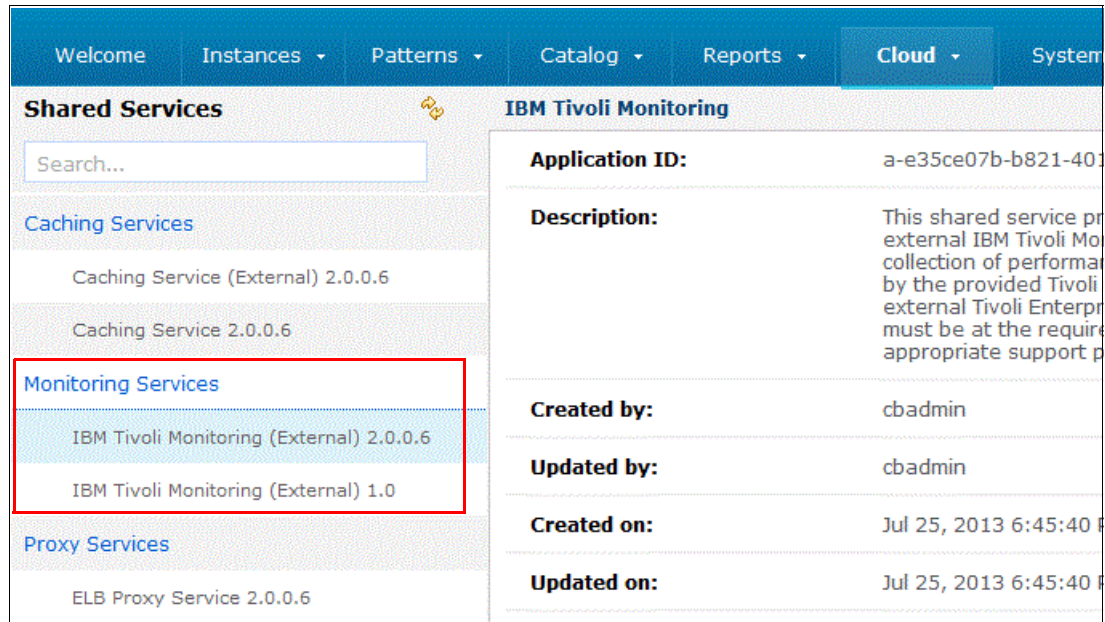
Key is controlling the virtualized infrastructure and gaining as much visibility as possible into the make-up of your private cloud. Monitoring things such as resource trends is fundamental to providing this control and visibility, with which you can be proactive in your environment resource management instead of putting out fires as they erupt.

Each cloud solution that is described in this Redpaper offers virtual application monitoring (see “Monitoring” on page 111). Workload Deployer and IBM SmartCloud Orchestrator offer the ability to integrate their systems monitoring functions with an external, existing IBM Tivoli Monitoring Infrastructure. PureApplication System goes even farther and provides a system-wide, fully integrated, and pre-configured monitoring infrastructure.



## 4.5.1 Integrating external monitoring systems

IBM cloud environments offer monitoring capabilities through the IBM Tivoli Monitoring (External) shared service, as shown in Figure 4-51.



Shared Services	
Search...	
Caching Services	
Caching Service (External) 2.0.0.6	
Caching Service 2.0.0.6	
Monitoring Services	
IBM Tivoli Monitoring (External) 2.0.0.6	
IBM Tivoli Monitoring (External) 1.0	
Proxy Services	
ELB Proxy Service 2.0.0.6	

IBM Tivoli Monitoring	
Application ID:	a-e35ce07b-b821-401
Description:	This shared service pr external IBM Tivoli Mo collection of perform by the provided Tivoli external Tivoli Enterpr must be at the require appropriate support p
Created by:	cbadmin
Updated by:	cbadmin
Created on:	Jul 25, 2013 6:45:40
Updated on:	Jul 25, 2013 6:45:40

Figure 4-51 IBM Tivoli Monitoring shared services

This shared service gives you the ability to use an external IBM Tivoli Monitoring Infrastructure to collect performance and availability information by using provided Tivoli Monitoring agents. Your external Tivoli Enterprise Monitoring infrastructure must be at version 6.2.2 Fix Pack 5 or later.

After it is created, the UNIX or Linux OS monitoring agents and the Workload monitoring agent that are provided for virtual application workloads are automatically connected to the defined instance of the Tivoli server. They use the information that was provided at deployment time regarding the primary and failover Tivoli Enterprise Management server, protocol, and port, as shown in Figure 4-52 on page 124.

Configure and deploy a shared service

Service name: IBM Tivoli Monitoring

▼ sharedservice - External Tivoli Enterprise Monitoring Server - default

\* Primary server: tems1.itso.ibm.com

Secondary server: tems2.itso.ibm.com

\* Protocol:

- ☒ IP.PIPE
- ☐ IP.SPIPE
- ☐ IP.UDP
- ☐ IP6.PIPE
- ☐ IP6.SPIPE
- ☐ IP6.UDP

\* Port: 1918

Console URL:

OK Cancel

Figure 4-52 Required information to deploy the external monitoring service

By using the shared service, you can consolidate monitoring in an enterprise console beyond the boundary of your private cloud.

After the system is deployed, you can access your Tivoli Enterprise Portal to start monitoring the private cloud environment, as shown in Figure 4-53.

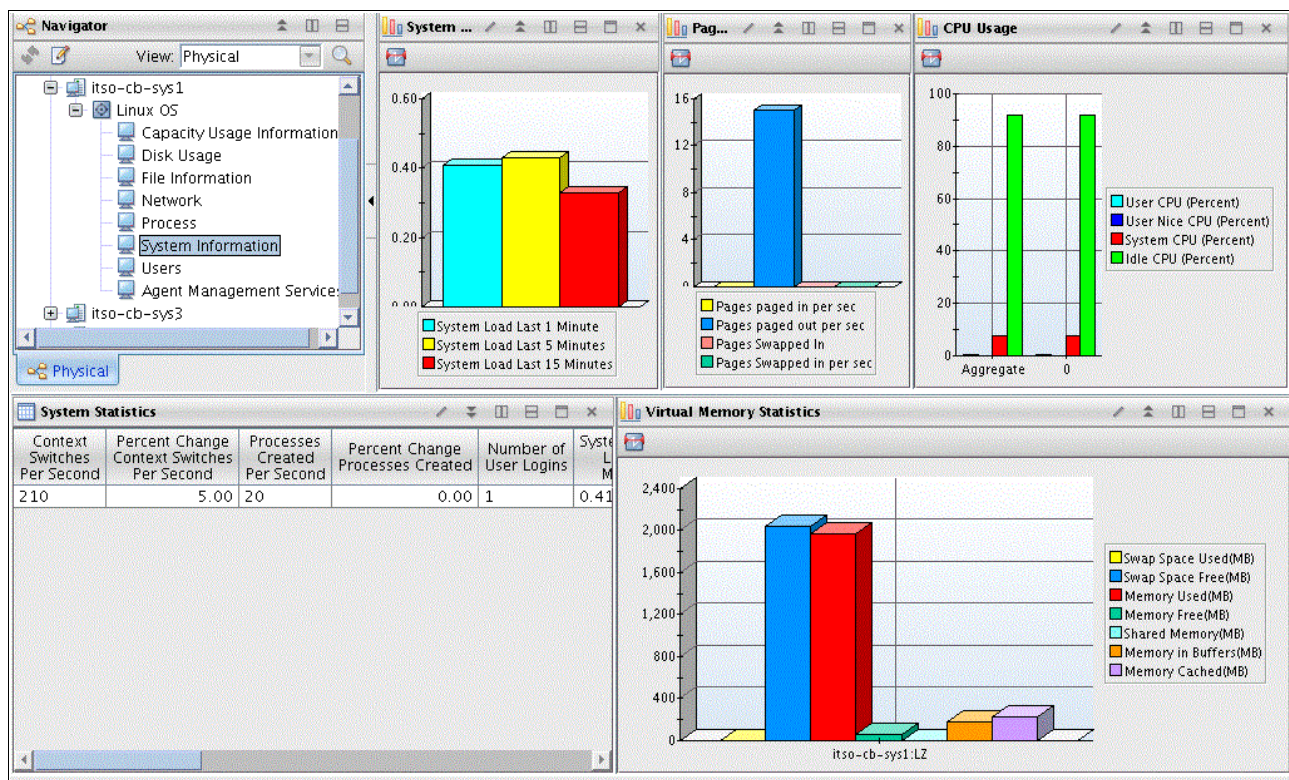


Figure 4-53 System data on Tivoli Enterprise Portal

For more information about the IBM Tivoli Monitoring (External) shared service, see these resources:

- IBM PureApplication System W1500 Information Center:  
[http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.puresystems.appsys.1500.doc%2Fiwd%2Fapc\\_monitoring.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.puresystems.appsys.1500.doc%2Fiwd%2Fapc_monitoring.html)
- The IBM Tivoli Monitoring shared service pdf:  
<https://www.ibm.com/developerworks/community/files/basic/anonymous/api/library/b6f53616-38e7-47ca-bc35-4cc465e5fcb7/document/dd431fb0-f8e1-43be-b0d4-b035755dd097/media>

## 4.5.2 Monitoring databases in IBM cloud technologies

Workload Deployer and SmartCloud Orchestrator each provide a set of tools for monitoring database performance.

Workload Deployer has a specific option in the Catalog menu to access these tools, as shown in Figure 4-54.

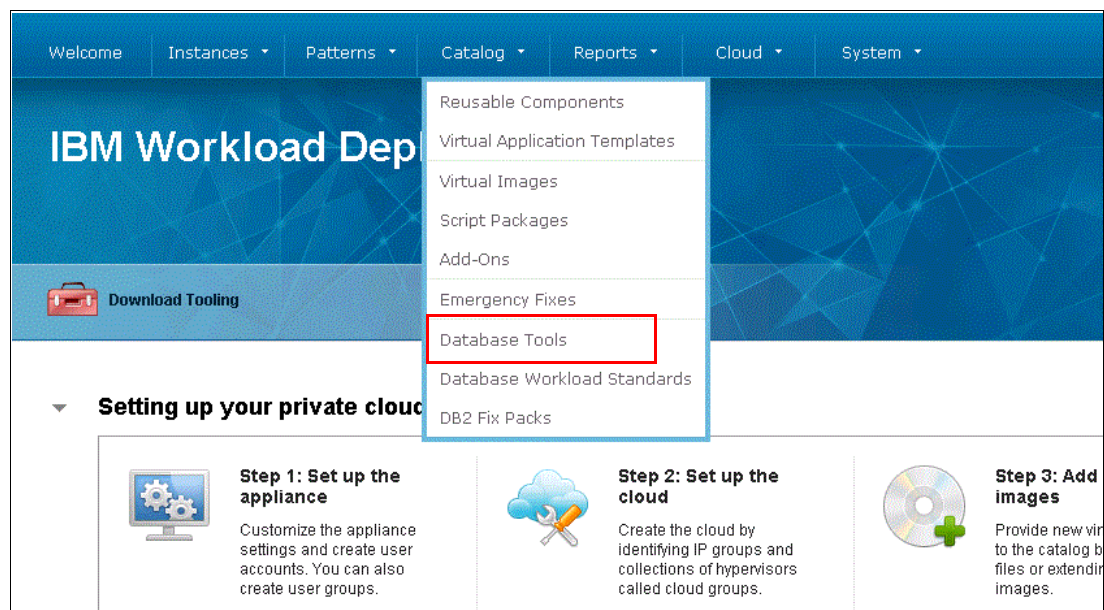


Figure 4-54 Accessing database monitoring tools in Workload Deployer

Workload Deployer and SmartCloud Orchestrator offers options to download the full IBM Data Studio client or InfoSphere® Warehouse SQL Warehousing tools (see Figure 4-55 on page 126). By using Workload Deployer and SmartCloud Orchestrator, you can deploy the Data Studio web console as a virtual application.



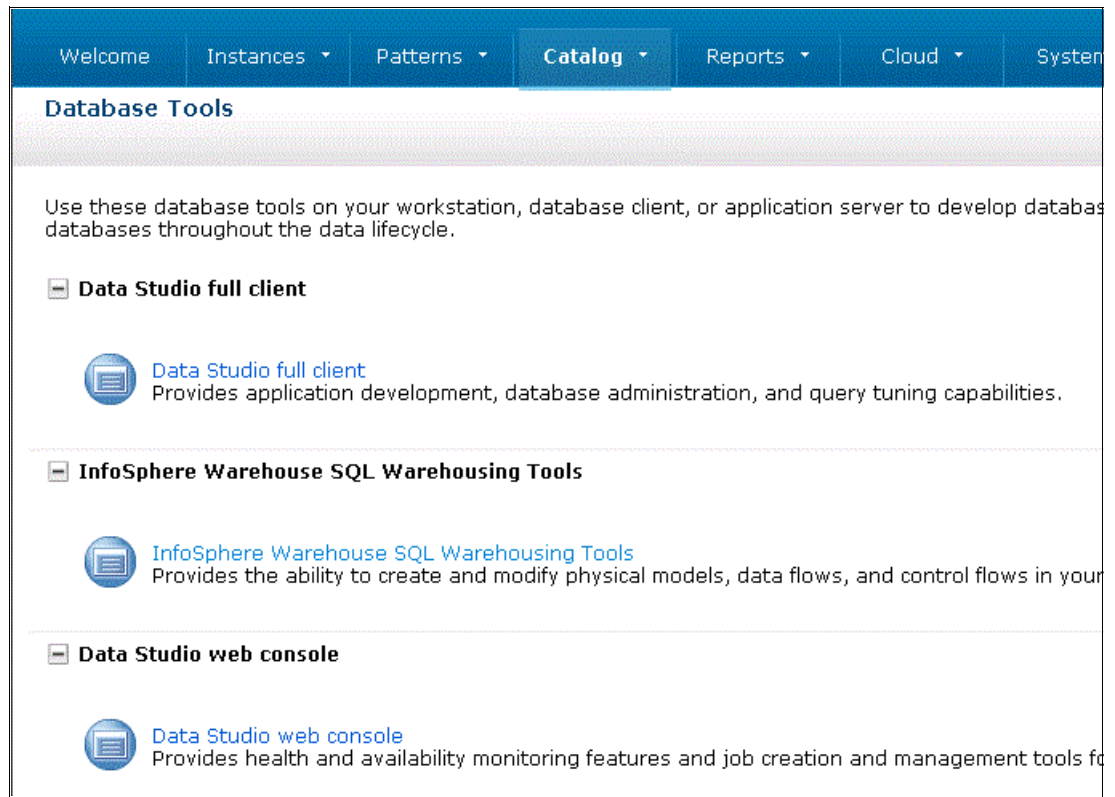


Figure 4-55 Database tools available in Workload Deployer

Deploying the Data Studio web console in Workload Deployer and SmartCloud Orchestrator requires that you first create a virtual application and deploy it. Complete the following steps to create and deploy the application and then use the web console to monitor databases:

1. From the Workload Console, go to **Pattern** → **Virtual Applications** and create a virtual application pattern by using Web Application Pattern Type 2.0. The Virtual Application Builder tool opens.
2. From the Application Components palette on the left side of the window, open Database Components and drag the Data Studio web console components on the canvas, as shown in Figure 4-56 on page 127. You must define the user name and password that is required to access the application.

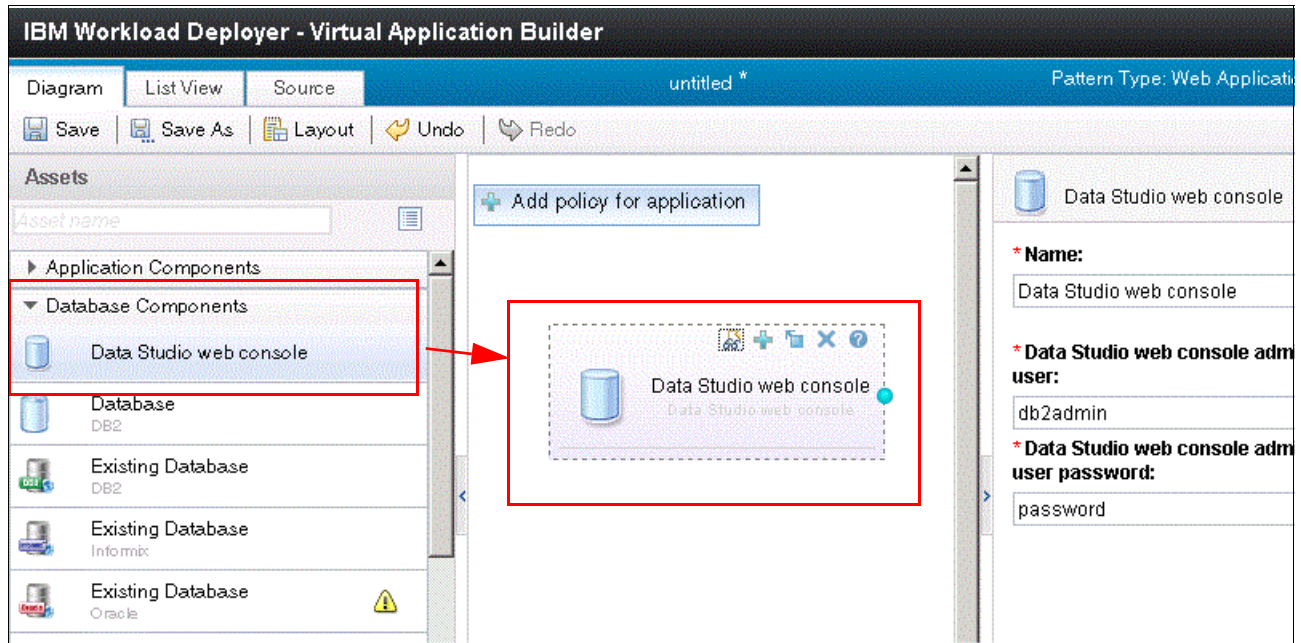


Figure 4-56 Virtual Application Builder to create Data Studio virtual application

3. Click **Save** to save the virtual application and close the Virtual Application Builder. You are redirected to the Virtual Application Pattern console (see Figure 4-57) and the listing for the virtual application you built. Now, you can deploy the Data Studio web application by clicking **Deploy**.

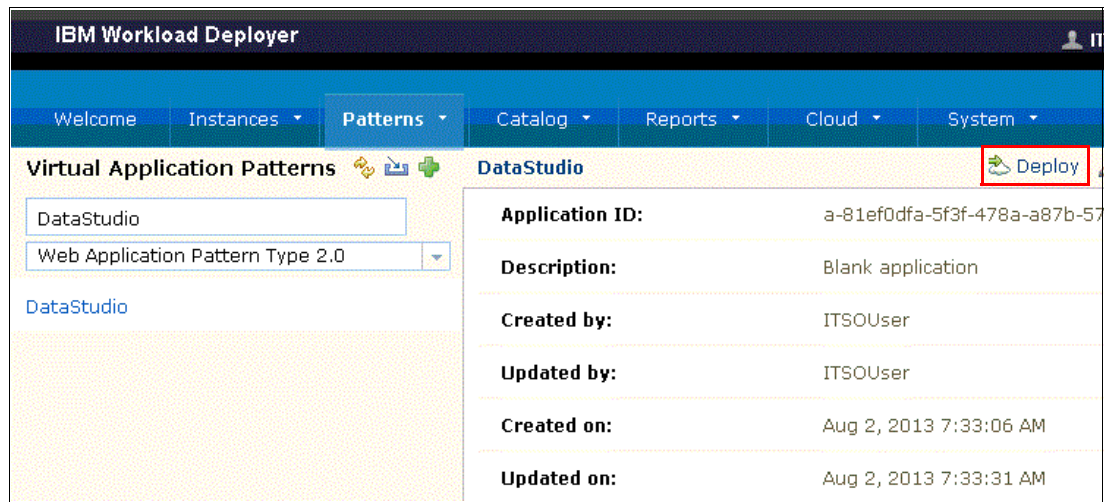


Figure 4-57 Deploying the Data Studio application

4. After the virtual application instance is deployed, you can access the Data Studio web console by clicking **Endpoint**, as shown in Figure 4-58 on page 128.

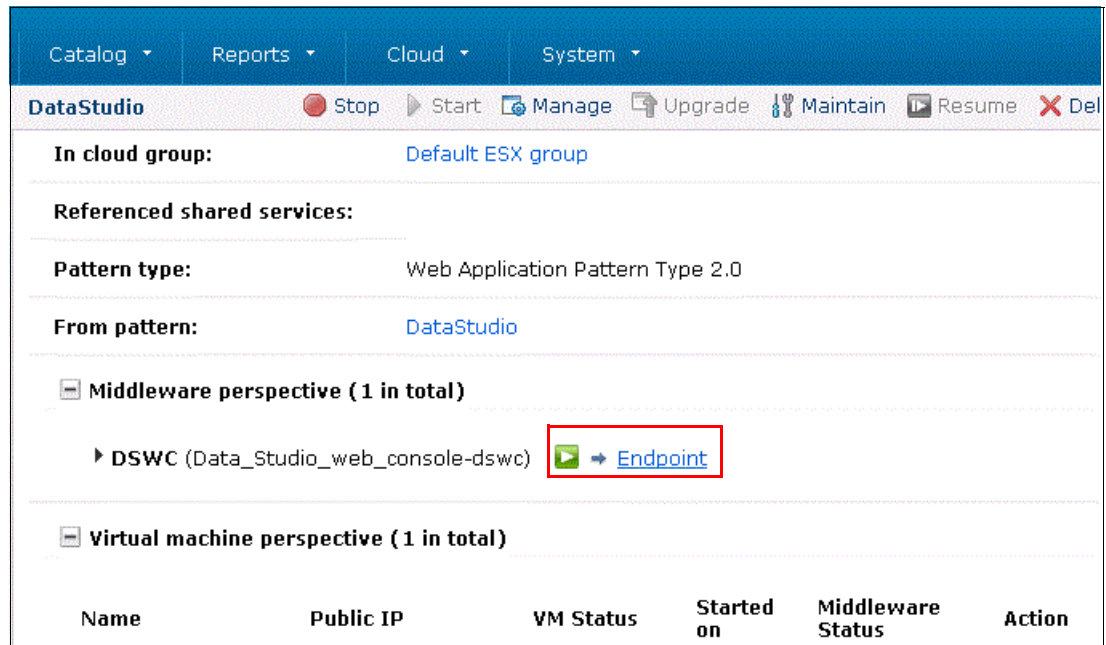


Figure 4-58 Use the Endpoint hyperlink to access the Data Studio console

5. Log in with the user name and password that were defined when you built the virtual application and you see the main console page, as shown in Figure 4-59.

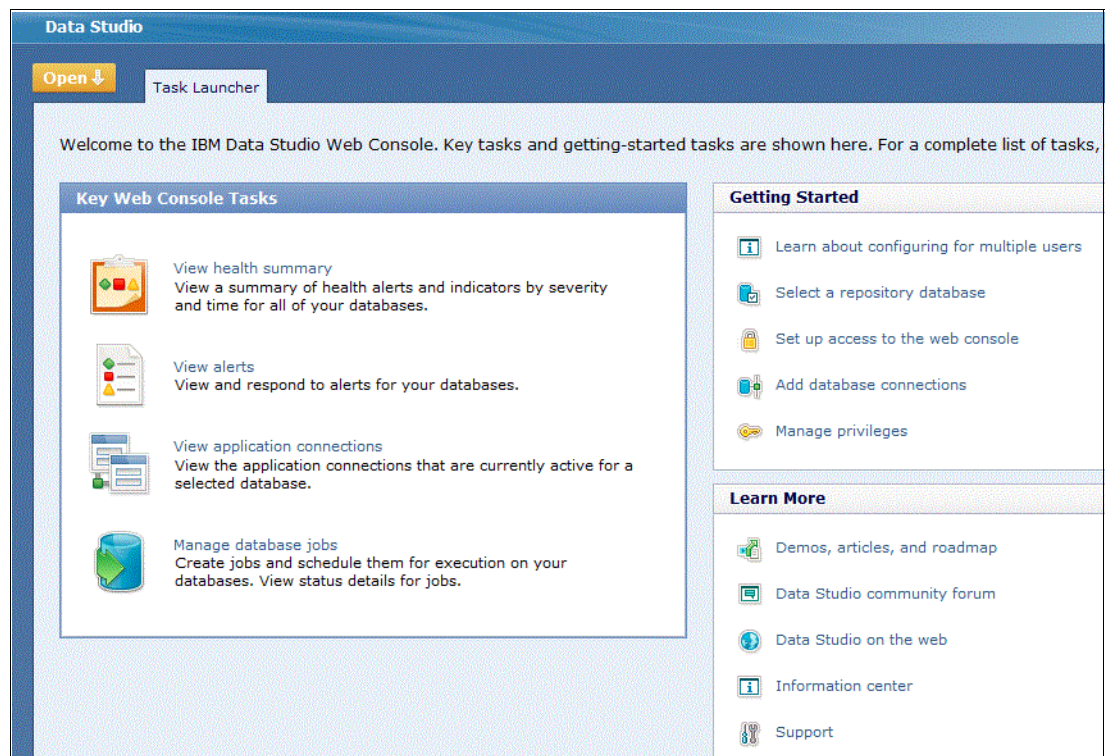
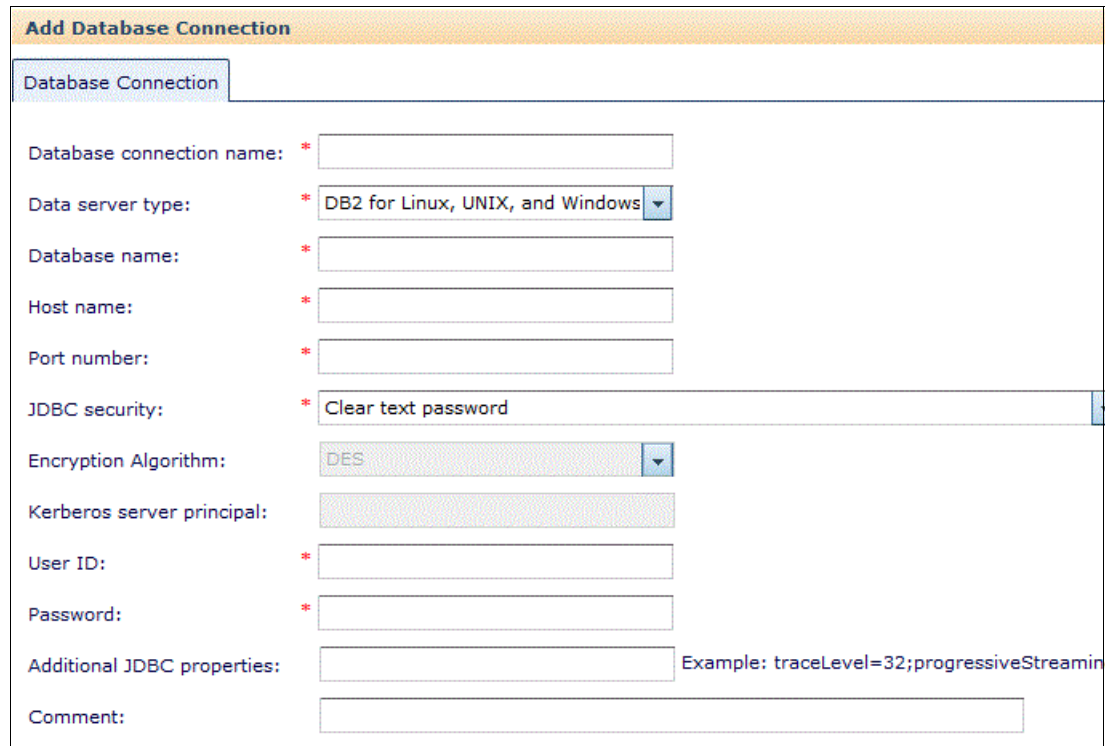


Figure 4-59 Data Studio web console



6. To start monitoring your database, you must add a database connection by entering the required information, as shown in Figure 4-60.

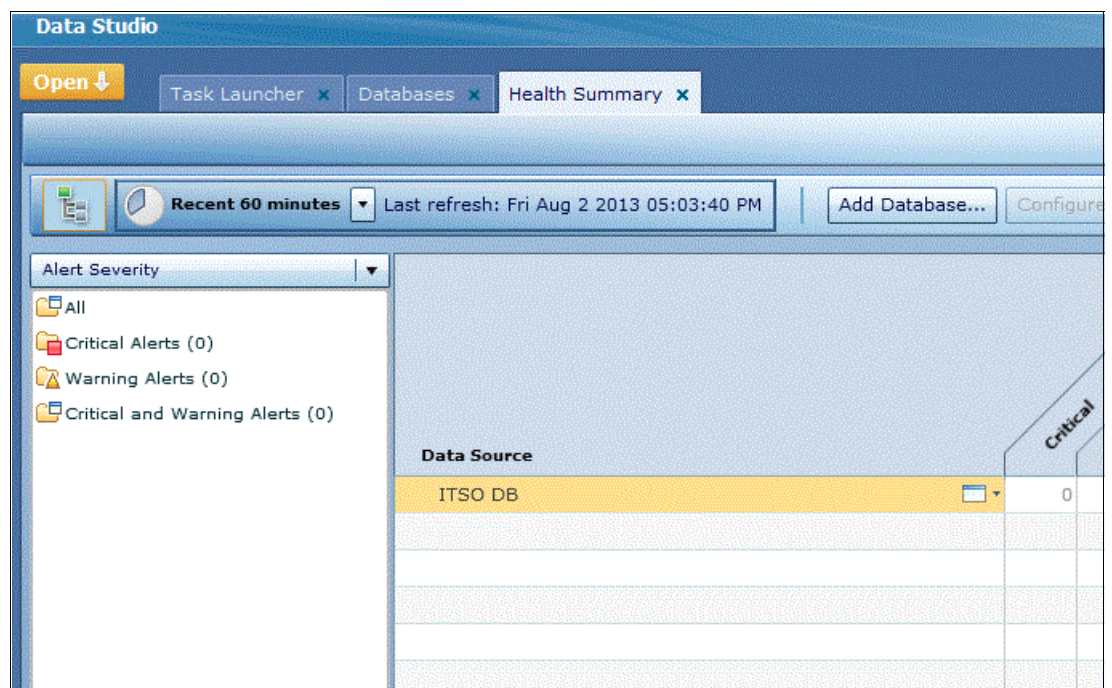


The 'Add Database Connection' form contains the following fields and options:

- Database connection name: \*
- Data server type: \* DB2 for Linux, UNIX, and Windows
- Database name: \*
- Host name: \*
- Port number: \*
- JDBC security: \* Clear text password
- Encryption Algorithm: DES
- Kerberos server principal:
- User ID: \*
- Password: \*
- Additional JDBC properties: Example: traceLevel=32;progressiveStreamin
- Comment:

Figure 4-60 Required information to add a database connection

7. After you add the database connection, you can connect to the database and start using the Data Studio web console to monitor it, as shown in Figure 4-61.



The Data Studio web console interface shows the following components:

- Open button
- Task Launcher, Databases, Health Summary tabs
- Recent 60 minutes dropdown, Last refresh: Fri Aug 2 2013 05:03:40 PM
- Add Database... button
- Configure button
- Alert Severity dropdown
- Alerts list: All, Critical Alerts (0), Warning Alerts (0), Critical and Warning Alerts (0)
- Data Source table with ITSO DB entry
- Critical alert indicator

Data Source	Alerts
ITSO DB	0

Figure 4-61 Monitoring the sample ITSO DB database from the Data Studio web console

For more information about how to use the Data Studio web console to monitor databases in Workload Deployer, see Chapter 11 of the IBM Redbooks publication *IBM Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud*, SG24-8011, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248011.html>

### 4.5.3 Using the integrated monitoring capabilities in PureApplication System

PureApplication System provides a system-wide, fully integrated and pre-configured monitoring infrastructure with which you can control and have visibility not only on the deployed system, but on each component in the full rack. This monitoring infrastructure is designed to make finding the necessary information as easy as possible for users and operators.

The PureApplication System Monitoring Portal offers a dashboard view of your critical enterprise IT resources, with which you can drill down into the individual elements of workloads. The interface is a Java based GUI that requires IBM Java 5 or higher.

The monitoring capabilities in PureApplication System are split into layers, with which you can monitor a system from its hardware components through the middleware to the running applications.

#### Hardware monitoring

PureApplication System provides an infrastructure map with which you can visually check the system's hardware components and their status. The map displays the compute nodes, system storage, network switches, and LED status.

To access the infrastructure map, start at the System Console and go to **Hardware** → **Infrastructure Map**, as shown in Figure 4-62.

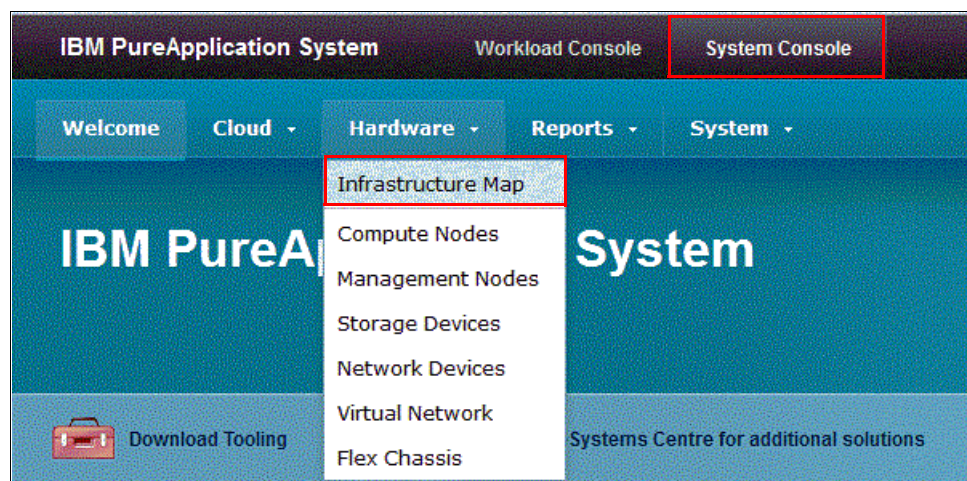


Figure 4-62 Opening the infrastructure map



Figure 4-63 shows the infrastructure map's default view for a 32-core PureApplication System environment. By using the different options, you can customize the view. To show different options in the map, select or clear the appropriate check boxes in the Legend pane on the left side of the window.

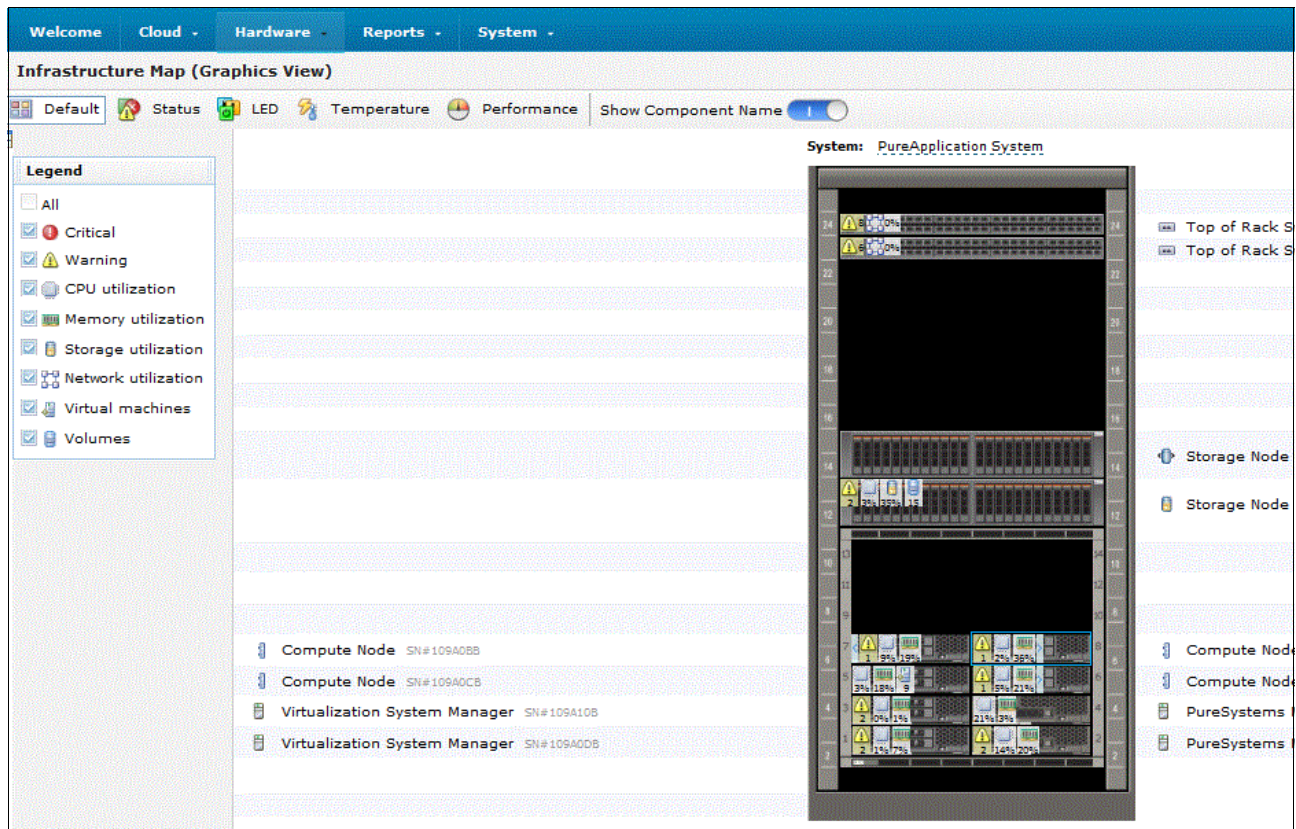


Figure 4-63 Infrastructure map (note Legend pane on left side of window)

Within the map area, clicking the listing for a specific system component displays detailed information about that component. Figure 4-64 shows the details about a compute node.

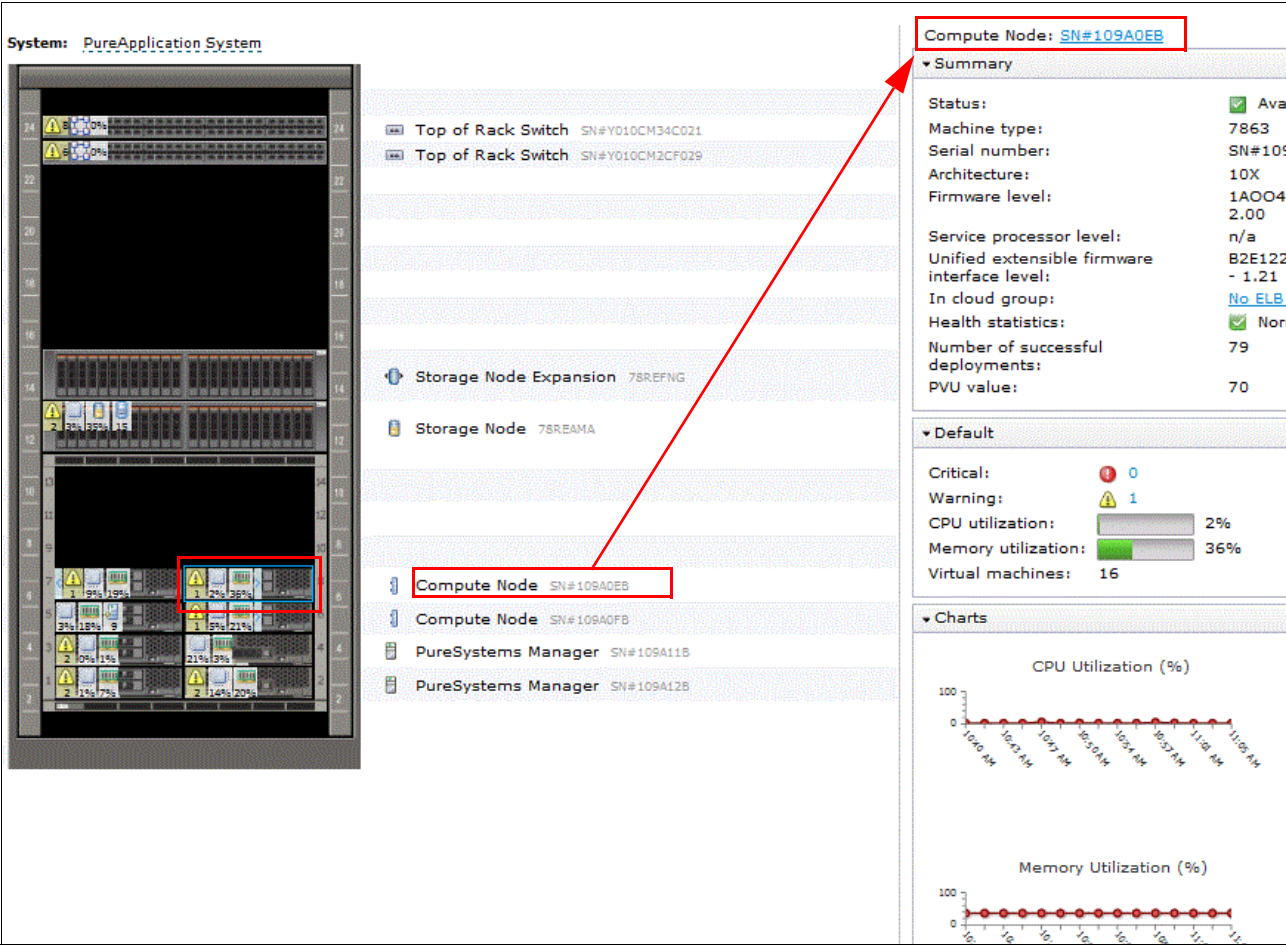


Figure 4-64 Viewing details about a specific compute node in the infrastructure map

Figure 4-65 shows the details of a storage node.

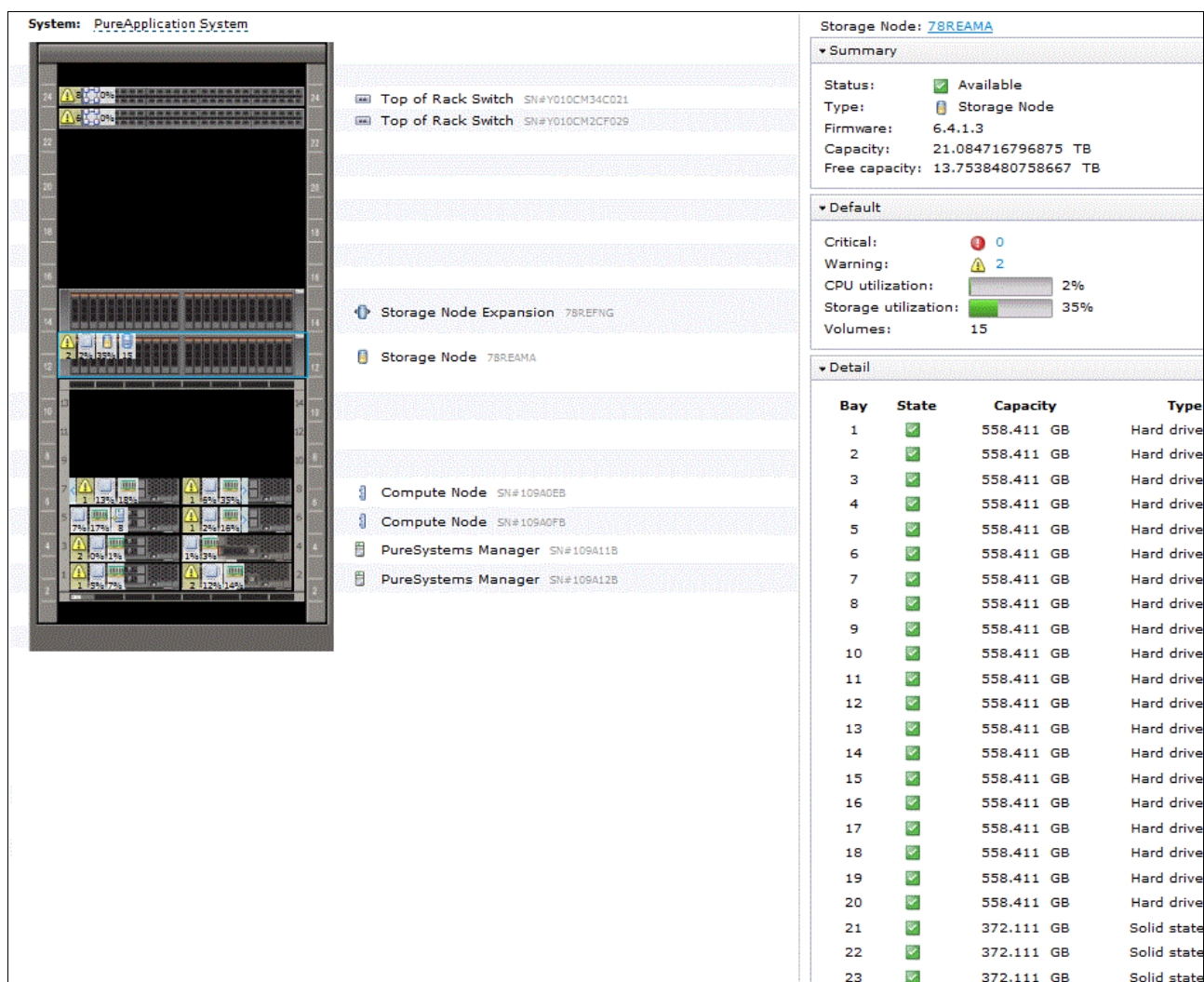


Figure 4-65 Viewing details about a specific storage node in the infrastructure map

An even more detailed view is available for each hardware component. This view is accessed from the Hardware menu (see Figure 4-63 on page 131) or by clicking the specific component link on the infrastructure map. Figure 4-63 on page 131 and Figure 4-64 on page 132 show this link at the top of the panel displaying the component information. Clicking the link takes you to the detailed component page that is shown in Figure 4-66 on page 134.



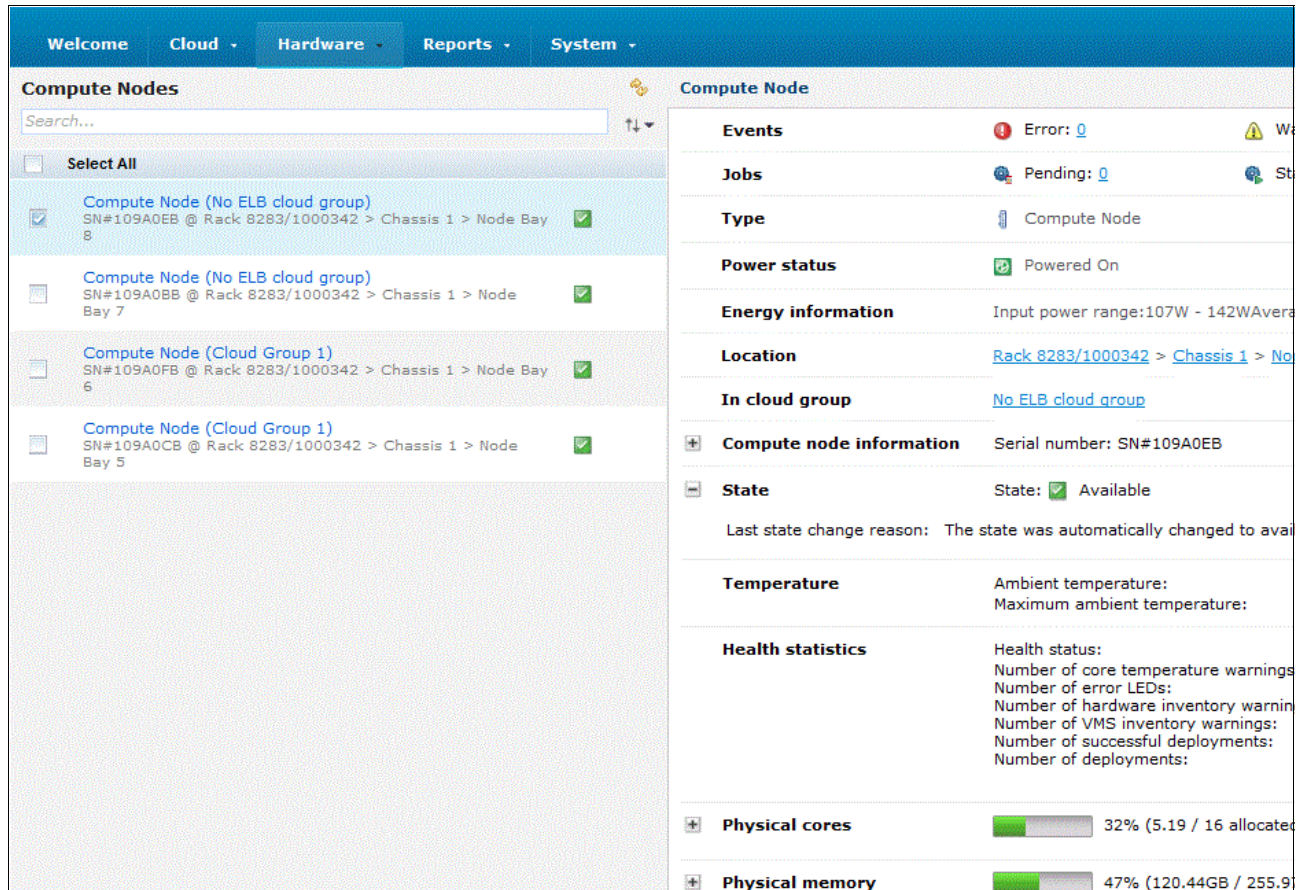


Figure 4-66 Detailed component page

If you have the appropriate permission, you can also administer the component from this page.

Another option to monitor the system is to use the Monitoring Agent for IBM PureApplication System. This agent can be deployed by deploying the System Monitoring shared service or by installing the agent on a separate computer, with which you can monitor the PureApplication System environment from an external IBM Tivoli Monitoring environment.

## Middleware monitoring

You also can monitor your system workloads and instances with PureApplication System. However, for the needed middleware monitoring facilities to be displayed in the Workload Console, you must deploy monitoring shared services. This is different from Workload Deployer and SmartCloud Orchestrator in that the monitoring infrastructure is deployed directly within the PureApplication System environment.

PureApplication System has different monitoring shared services that you can deploy to gain higher visibility to and insight about deployed middleware. These services are shown in Figure 4-67 on page 135 and described next.

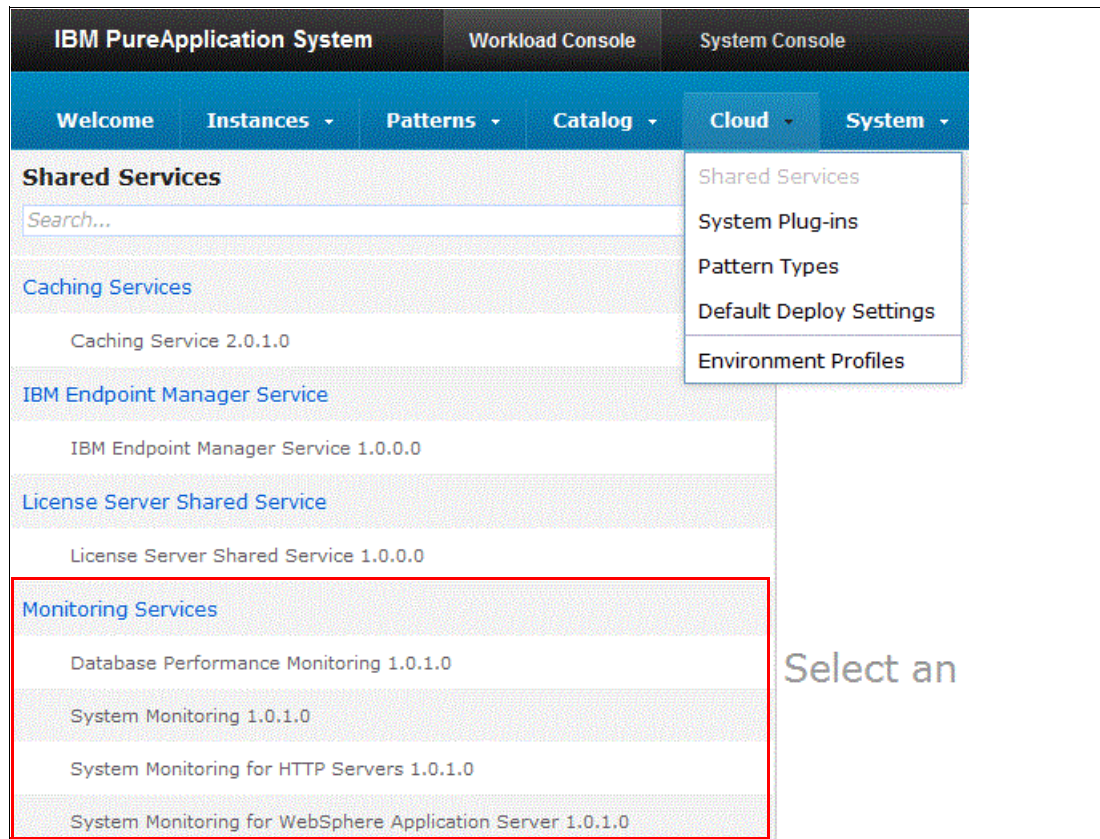


Figure 4-67 Monitoring shared service on IBM PureApplication System

The following monitoring shared services are available:

- ▶ **Database Performance Monitoring**  
This shared service provides the monitoring infrastructure that is needed to collect performance and availability information for DB2. With this information, you can troubleshoot your databases and make business decisions regarding your service usage.
- ▶ **System Monitoring**  
This shared service provides the monitoring infrastructure that is needed to collect performance and availability information by the provided monitoring agents. With this information, you can troubleshoot your system and make business decisions regarding your hardware and service usage.
- ▶ **System Monitoring for HTTP Servers**  
This shared service can extend the System Monitoring infrastructure with monitoring agents for HTTP Servers.
- ▶ **System Monitoring for WebSphere Application Server**  
This shared service can extend the System Monitoring infrastructure with monitoring agents for WebSphere Application Server.

### ***Monitoring database performance***

To use the Database Performance Monitoring shared service, you must be a cloud or appliance user. The service is used to isolate and analyze typical database performance problems, and you can check the health of your database and drill down for more details.

The Database Performance Monitoring shared service provides the following functions:

- ▶ An individual health indicator for each database that is monitored to inform you about whether all monitored aspects of the database are running as expected.
- ▶ A set of other indicators to identify which part of a particular database is experiencing performance problems. The indicators cover the following areas of performance:
  - I/O
  - Locking
  - Logging
  - Memory
  - Recovery
  - Sorting
  - Storage
  - Workload
  - Data server status
- ▶ Monitoring links that provide quick access to a health summary overview of each database. You also have access to the following dashboards:
  - SQL Statements
  - Buffer Pool and I/O
  - Locking
  - Logging
  - Memory
  - Utilities
  - Workload dashboards
  - Connection dashboards

Some of the dashboards include the ability to call services to tune database performance. For example, the SQL Statements dashboard can start Optim™ Query Workload Tuner to analyze and improve the performance of slow-running queries and statements.

- ▶ Real-time monitoring and performance warehousing, with automated aggregation and retention management. Dashboards can display any collected data, which makes it easy to review metrics for older periods, compare current values to older ones, and view metrics trends. Predefined reports are available for selected performance metrics, including analyzing high-cost SQL and viewing disk consumption and growth.
- ▶ Access (for administrative users) to multi-database performance monitoring for all database instances that are deployed in a cloud group. The service offers several dashboards that provide information about monitored databases in a cloud group. The Databases dashboard displays the list of monitored databases. The Health dashboards display health summaries, alerts, current application connections, current table spaces, and current utilities. Some dashboards, such as Health and Alerts, display information for all databases while other dashboards require the administrative user to choose a specific database to examine.

The Database Performance Monitoring shared service must be up and running for the monitoring services to be available. Other monitoring service (beyond those that were previously described), can be accessed by clicking **Manage** next to a particular database instance. By using these services, you can view health indicators, start, stop, and restart monitoring (for a whole storage system or an individual database), and apply emergency fixes to the monitoring shared service.

To access the Database Performance Monitoring shared service in PureApplication System, you deploy the shared service. After the service is deployed, you can access the monitoring console by clicking **Endpoint** on the shared service pane, as shown in see Figure 4-68 on page 137.



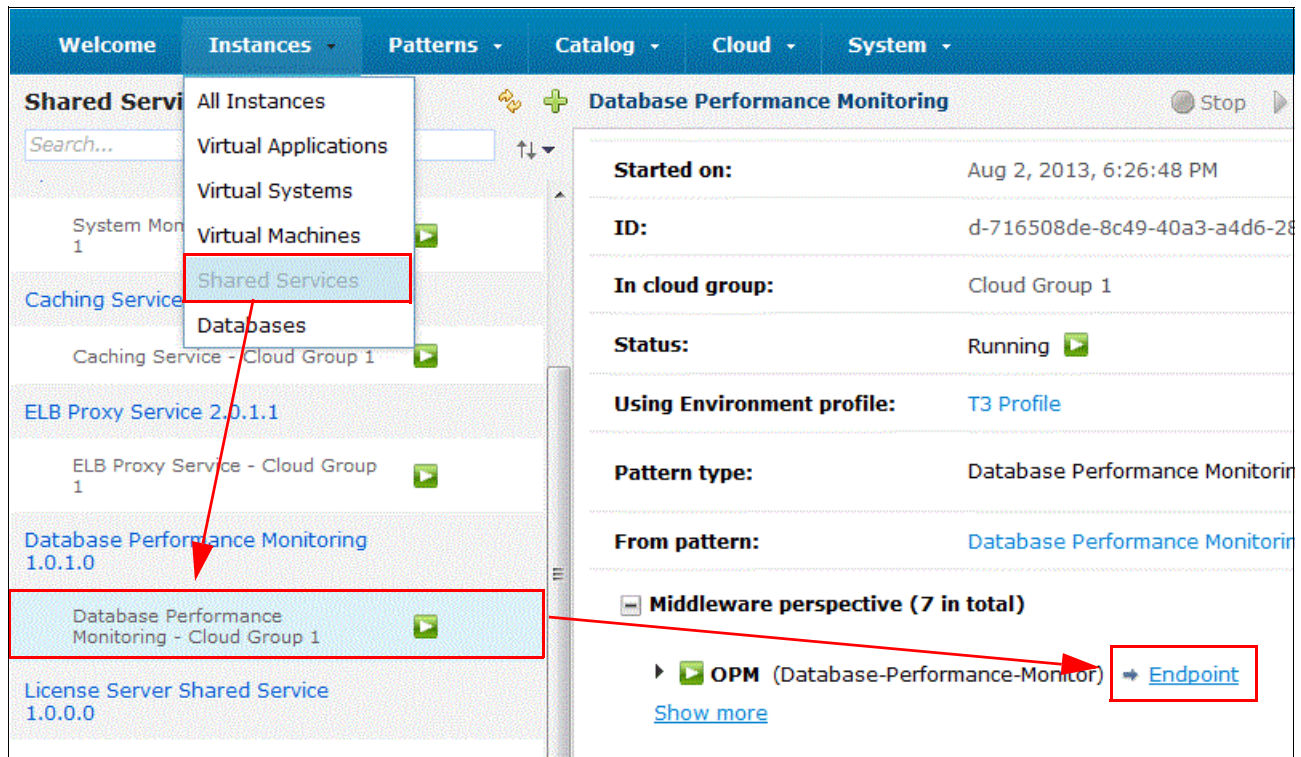


Figure 4-68 Accessing the Database Performance Monitoring shared services monitoring console

When the console is opened, all of the databases within the cloud group or in the same environment profile are automatically displayed. Figure 4-69 shows the console with the databases running in the same environment profile as the monitoring services.

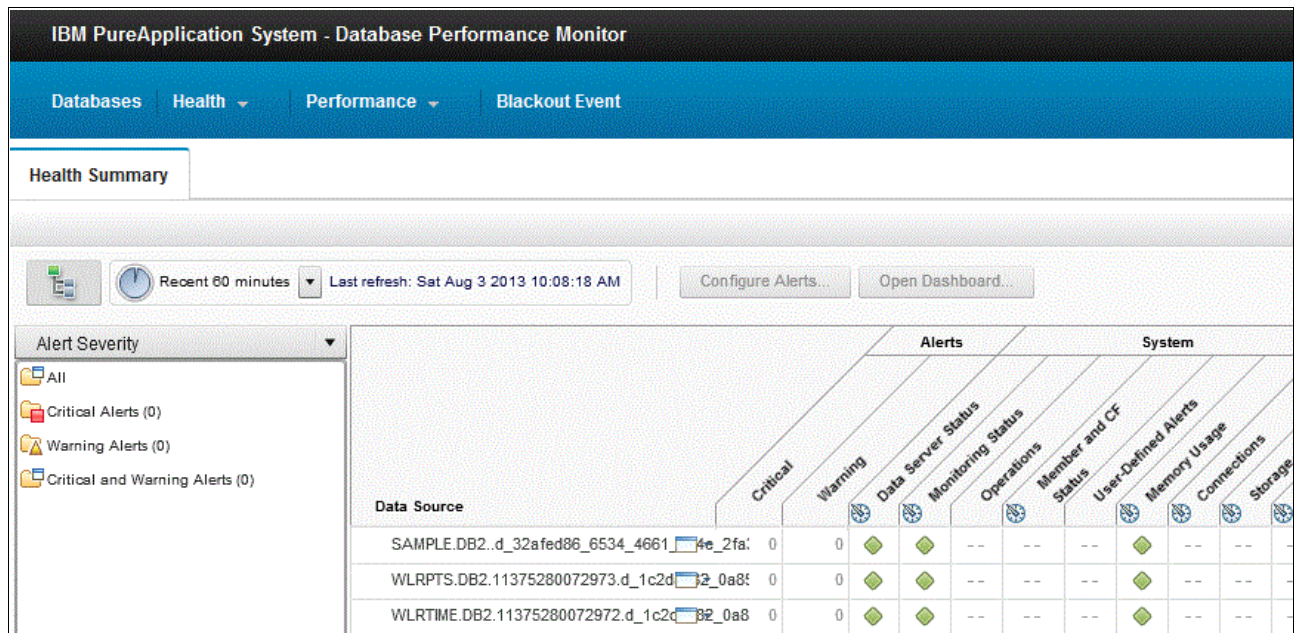


Figure 4-69 Database Performance Monitoring shared services monitoring console

If a performance issue arises in any monitored database, the console displays a graphical depiction of the problem at the next refresh cycle, as shown in Figure 4-70.

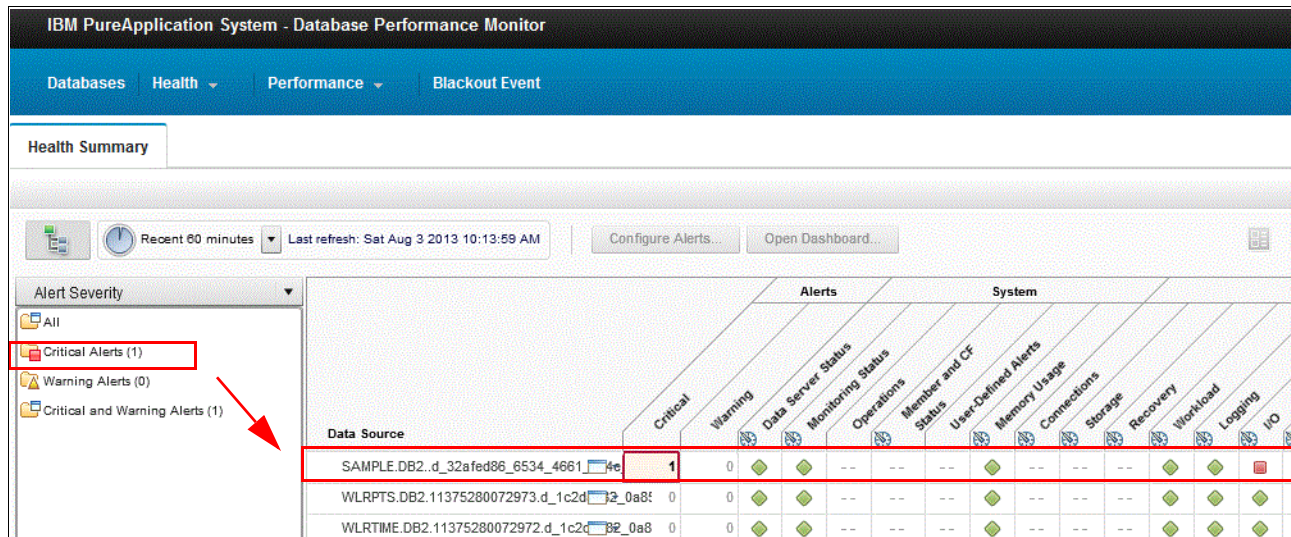


Figure 4-70 Graphical depiction of database performance, including an alert (see IO column on right of window)

The example that is shown in Figure 4-70 shows a small red square in the IO column on the right side of the window. Clicking the red square calls up more information in which you can drill down for analysis and troubleshooting, as shown in Figure 4-71.

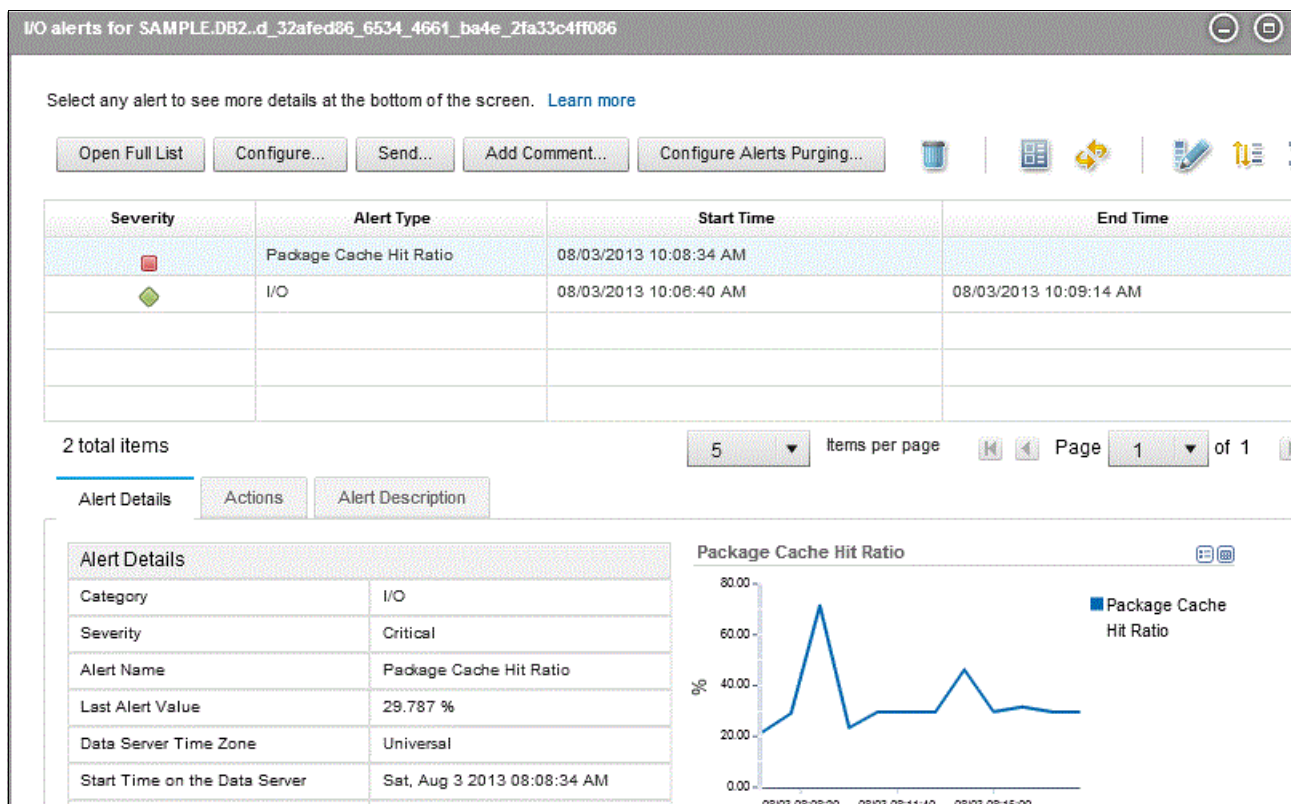


Figure 4-71 More information about the I/O alert



All of available performance analysis tools are listed in the Performance menu, as shown in Figure 4-72.

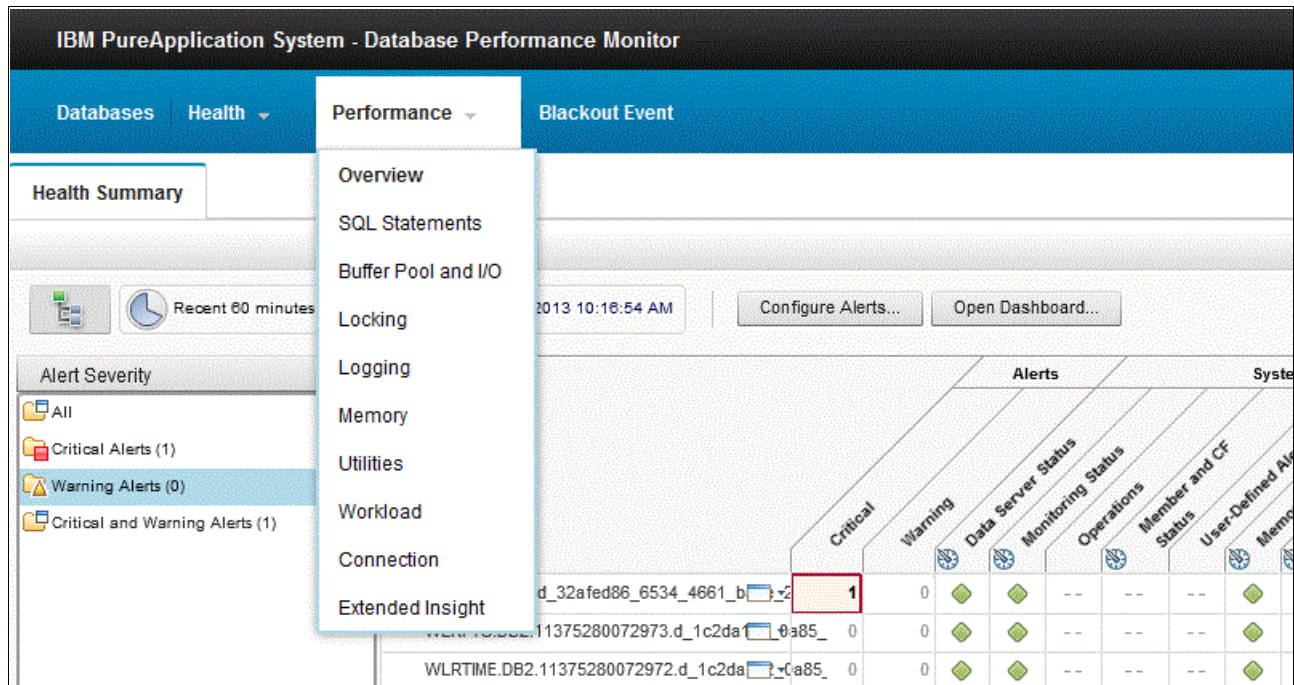


Figure 4-72 Performance analysis tools

For more information about these database monitoring functions, see Chapter 11 of the IBM Redbooks publication *IBM Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud*, SG24-8011, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg248011.html>

You can also see the IBM PureApplication System W1500 Information Center at this website:

[http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/topic/com.ibm.puresystems.appsys.1500.doc/OPM/com.ibm.datatools.perfmgmt.monitor.doc/p\\_monitor.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/topic/com.ibm.puresystems.appsys.1500.doc/OPM/com.ibm.datatools.perfmgmt.monitor.doc/p_monitor.html)

### **Monitoring WebSphere Application Server performance**

To monitor WebSphere Application Server, you deploy the System Monitoring for WebSphere Application Server shared service. By using this service, you can monitor WebSphere Application Server versions 7, 8, and 8.5, even if you have multiple installations on the same physical node.

The monitoring agent for this service collects the following types of data through the data collector that is embedded in the WebSphere Application Server process:

- ▶ Application server requests from the ITCAM Data Collector for WebSphere
- ▶ Resource data from WebSphere Performance Monitoring Infrastructure (PMI)
- ▶ Data from WebSphere log files
- ▶ Process data from the operating system

When you start the System Monitoring Portal, you can select from the drop-down menu the system that you want to monitor, as shown in Figure 4-73.

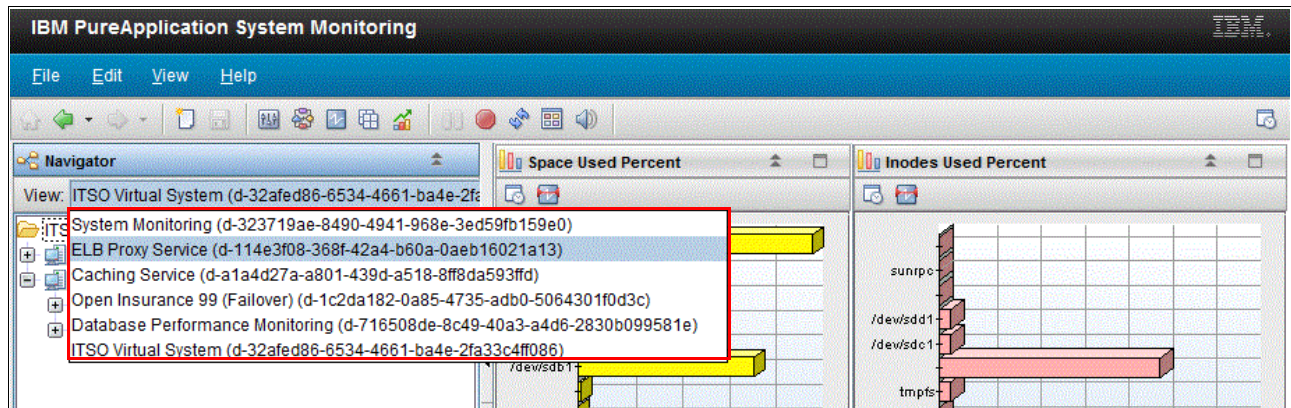


Figure 4-73 Selecting a system to monitor

With the system selected, you then drill down to analyze its performance by using different agents. Figure 4-74 shows the available Linux OS agent view.

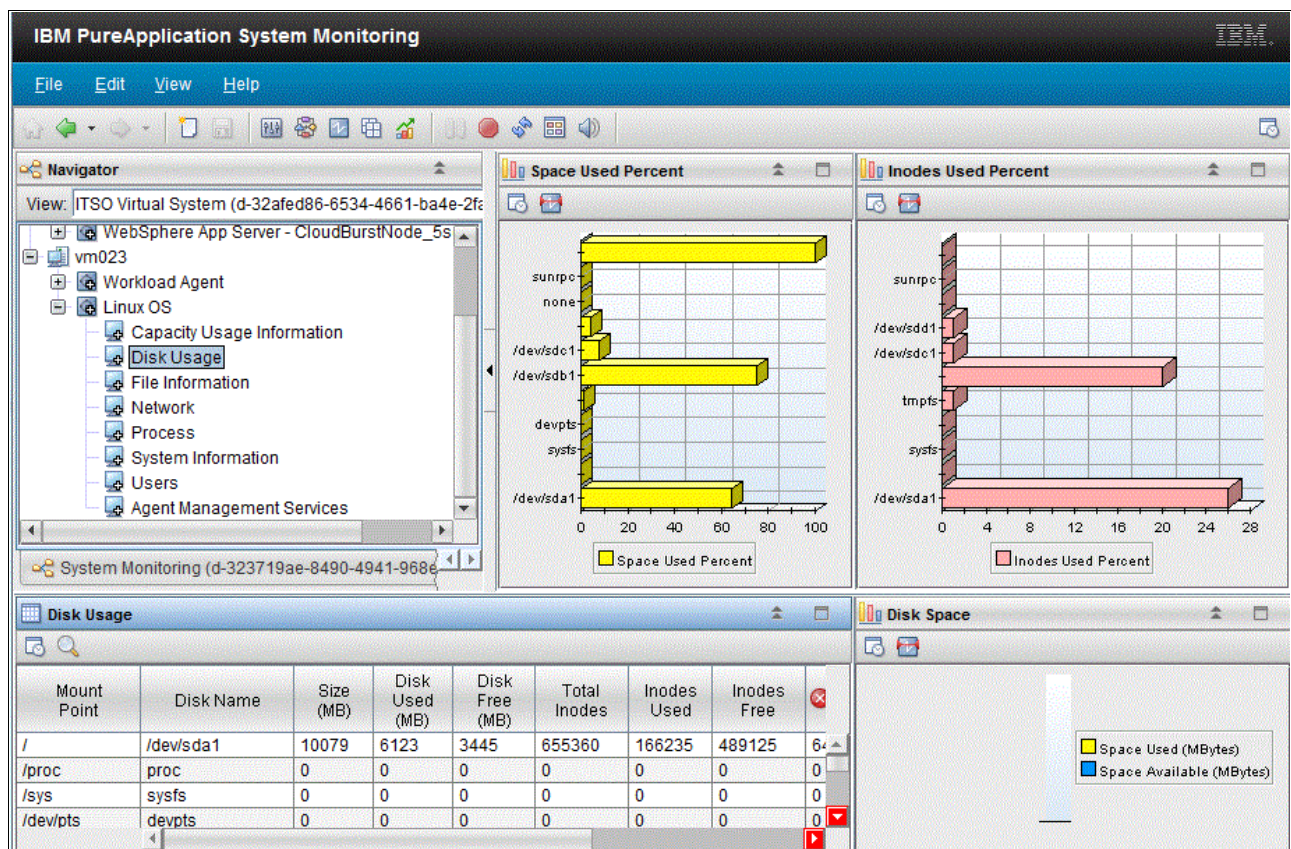


Figure 4-74 Linux OS agent view



Figure 4-75 shows the WebSphere Application Server agent view.

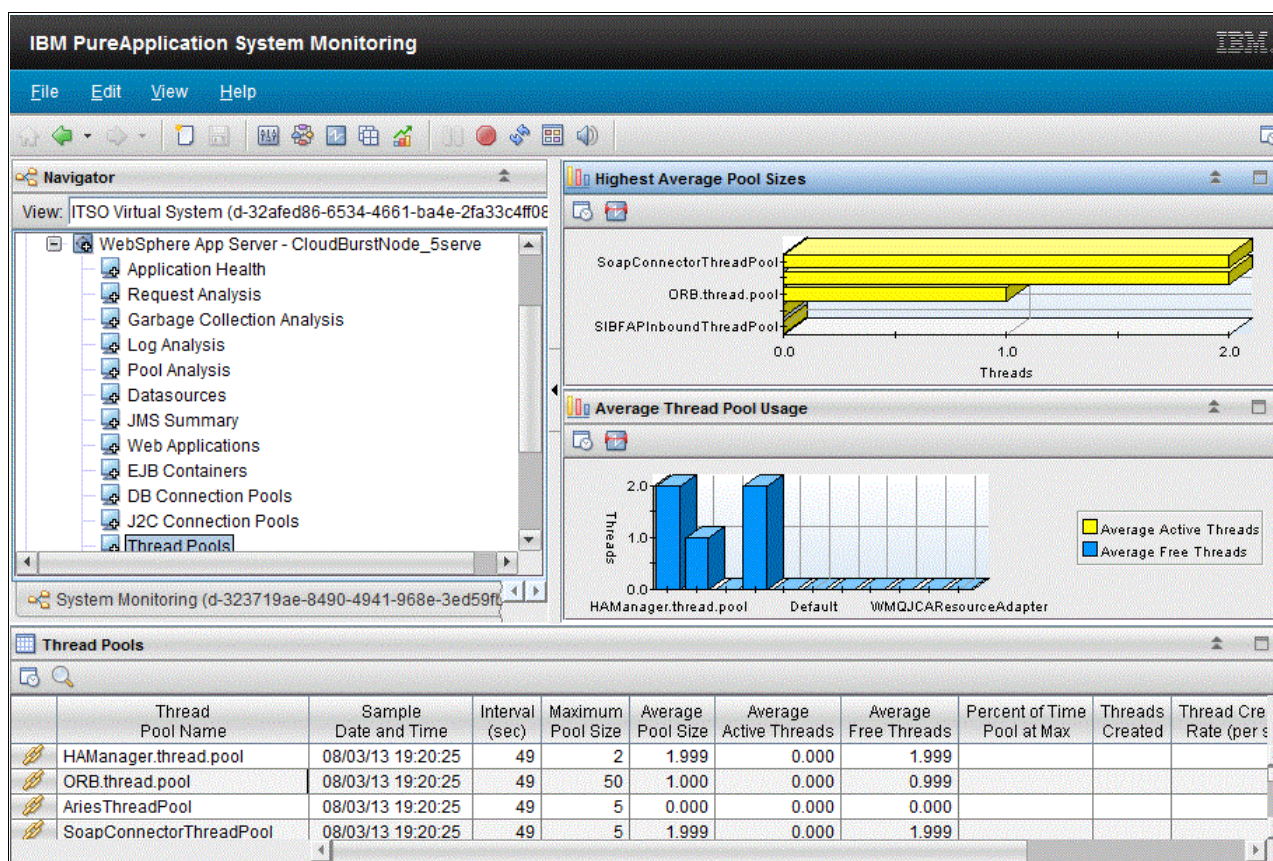


Figure 4-75 WebSphere Application Server agent view

For more information about monitoring middleware, see the IBM PureApplication System Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fwd%2Fvit\\_maninsta.html](http://pic.dhe.ibm.com/infocenter/psappsys/v1r0m0/index.jsp?topic=%2Fcom.ibm.ipas.doc%2Fwd%2Fvit_maninsta.html)

## 4.6 Pattern management and continuous delivery

As described in 4.2, “Virtual system lifecycle” on page 87, virtual system patterns and virtual application patterns are composed of different artifacts. In this section, we describe the deployment and upgrade lifecycle of these components.

### 4.6.1 Virtual system development and continuous delivery

Any component in a virtual system pattern can be the object of development and improvement efforts. Changes in any components must be carefully managed to avoid uncontrolled situations.

The typical virtual system components are virtual images and script packages.

## Virtual image management

One of the biggest challenges in managing a virtualized environment is so-called image sprawl. Image sprawl occurs when you create an image because you do not know, or have limited information about, the images that are already available. In a sprawled environment, resizing and standardizing your image catalog can be a significant problem.

IBM cloud technologies, such as Workload Deployer and SmartCloud Orchestrator, provide the following tools to help avoid the problems that are associated with image sprawl:

- ▶ IBM Image Construction and Composition Tool (ICCT)
- ▶ Script packages
- ▶ The catalog where you store your images and script packages
- ▶ The virtual image library (only available in SmartCloud Orchestrator)

Image management typically sits between the following two extremes:

- ▶ Use a base OS image for everything and then customize it with scripts or by manually installing software components
- ▶ Create an image for everything and eventually experience image sprawl

The advantages of creating a virtual image with the required software preinstalled and pre-configured are clear. So it makes sense to have virtual images with more than just the base OS installed. However, there is no common recipe for the perfect balance of image customization; it depends on individual organizational requirements.

Consider the following rules when you are determining the right balance in your image catalog:

- ▶ If you need a software component in every instance of the image that you deploy, it is probably advantageous to have the software component in the image.
- ▶ If the size of software component slows down the copy on the image and the installation and configuration processes take a long time, it is probably a good idea to have the software component in the image.
- ▶ If you change the software component frequently, it is probably a good idea to script the changes.

Because every company is unique and has its own way of maintaining virtual images, you can add your own considerations to this list.

The use of ICCT and script packages fits well with these considerations, so you should consider their use to limit your image catalog size.

For more information, see the developerWorks article, *Defeat image sprawl, once and for all*, which is available at this website:

[http://www.ibm.com/developerworks/websphere/techjournal/1112\\_col\\_willenborg/1112\\_col\\_willenborg.html](http://www.ibm.com/developerworks/websphere/techjournal/1112_col_willenborg/1112_col_willenborg.html)

## Virtual system configuration management

Another important topic when you are working with virtual systems is managing the system configurations. This refers to the following considerations:

- ▶ Saving a copy of your system configuration so that you can destroy it and redeploy the system a second time. If you save a copy of the configuration, you can easily apply it back to restore your system.
- ▶ Promoting a configuration from one stage to another (for example, from the test to the pre-production to the production environment).

Saving a copy of your configuration can be done easily with a configuration management tool, such as Rational Automation Framework (RAF) or the Advanced Middleware Configuration (AMC) tool, which comes pre-entitled for WebSphere Application Server on PureApplication System.

The AMC tool is RAF provided as a virtual system image. You can deploy the AMC tool by creating a pattern and selecting the appropriate image, as shown in Figure 4-76.

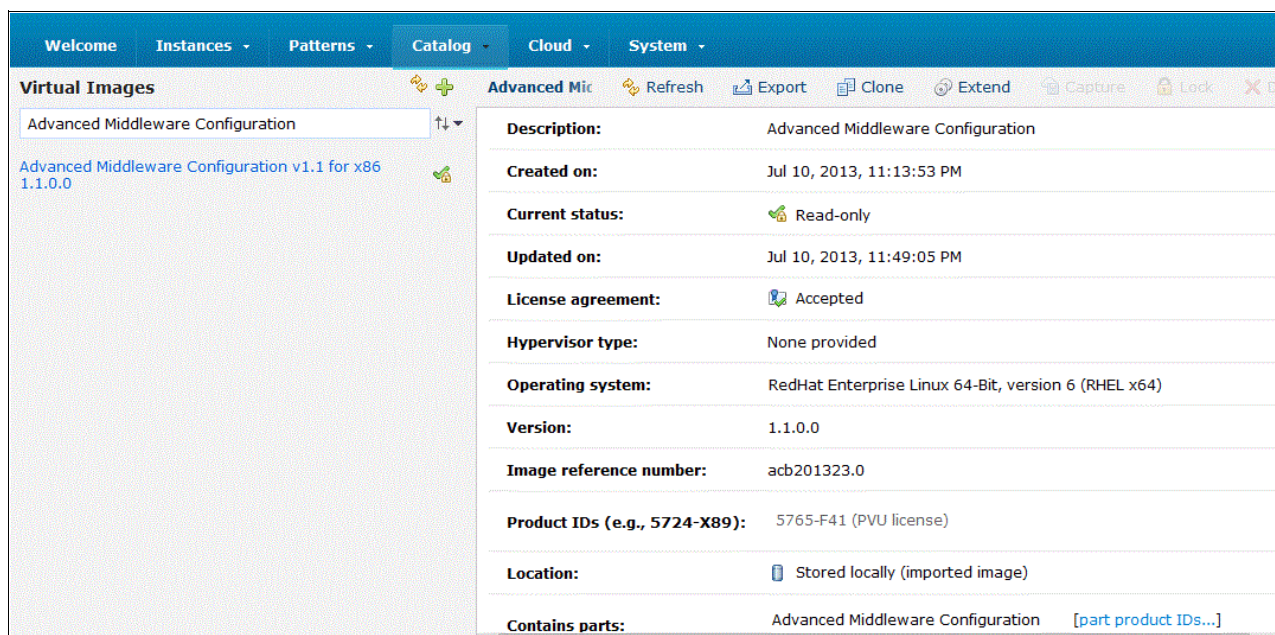


Figure 4-76 Advanced Middleware Configuration virtual image

By using the AMC tool and RAF, you can capture a virtual system configuration (it can also be a system that is running outside of IBM PureApplication System) and automatically create a virtual system that reflects the system that is captured. This can be a useful function when you bring your systems onto IBM PureApplication System.

Configurations in the AMC tool and RAF can be versioned, so you can return to a previous configuration, if needed.

For more information about these tools, see the following articles:

- IBM developerWorks article *Understanding IBM Rational Automation Framework*:  
<http://www.ibm.com/developerworks/rational/library/understanding-ibm-rational-automation-framework/>
- IBM developerWorks article *WebSphere CloudBurst plus Rational Automation Framework for WebSphere*:  
<http://www.ibm.com/developerworks/cloud/library/cl-hardinfra/>
- IBM developerWorks article *Preparing for IBM PureApplication System, Part 4: Onboarding applications to the cloud using the Advanced Middleware Configuration tool V1.1*:  
[http://www.ibm.com/developerworks/websphere/library/techarticles/1204\\_akeley/1204\\_akeley.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1204_akeley/1204_akeley.html)

## Topology management

Topologies might need to be changed based on your evolving needs. The biggest issue in a topology change often is how you configure the middleware. The use of script packages, RAF, and the AMC tool can be used to help with topology management.

While RAF and AMC provide specific functions for topology changes, script packages must be carefully built and packaged so there is no effect if the topology changes.

### 4.6.2 Virtual application development and continuous delivery

A virtual application pattern consists of plug-ins that define components, links, policies, and configuration files. You can create or modify these parts by using the Plug-in Development Kit that was described in “IBM Workload Plug-in Development Kit” on page 37, and is provided with any IBM cloud technology offerings that are described in this book.

As with a virtual system pattern, a virtual application pattern also relies on a virtual image. However, unlike a virtual system pattern, if you are working with virtual application pattern, you are unlikely to face issues that are related to image sprawl because the image that is used when a virtual application pattern is deployed is a base OS image that is unique to every deployment (unless you change the image, as described in 4.3.1, “Base operating system image lifecycle” on page 106).

When you are dealing with a virtual application pattern, a change can affect a plug-in, with a new version released, or the topology, with a new component added to your virtual application in the virtual application builder.



## Pattern portability

The ability to use patterns in different cloud technologies and systems is called *pattern portability*, and it is an important consideration when applications are deployed to a cloud. Customers want to know that the systems that make up their patterns are deploying consistently, regardless of the cloud technology that is used.

By using pattern portability, customers can easily move between different types of clouds, such as from a public cloud to a private one. It also enables developers to introduce a useful level of abstraction above the provisioning and orchestration technologies.

This chapter includes the following topics:

- ▶ Standard pattern package formats
- ▶ Import and export prerequisites
- ▶ Exporting and importing patterns
- ▶ Portability example that uses IBM SCAS and PureApplication System
- ▶ Leading practices



## 5.1 Standard pattern package formats

Industry-wide standards for cloud and pattern portability are still being adopted. IBM is working with independent software vendors and the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) working group to further define standards for pattern or service definition adoption. The aim of the standards is to ensure portability between different cloud vendors and technologies and eventually create a vendor-neutral cloud ecosystem.

TOSCA defined a declarative model that enables software applications to be deployed consistently on virtual and physical infrastructures. The TOSCA standards enable the following important capabilities:

- ▶ Portability and interoperability of cloud services
- ▶ Model-driven cloud service management
- ▶ An appstore for cloud services
- ▶ Open, hybrid clouds

Figure 5-1 shows the TOSCA model at the time of this writing.

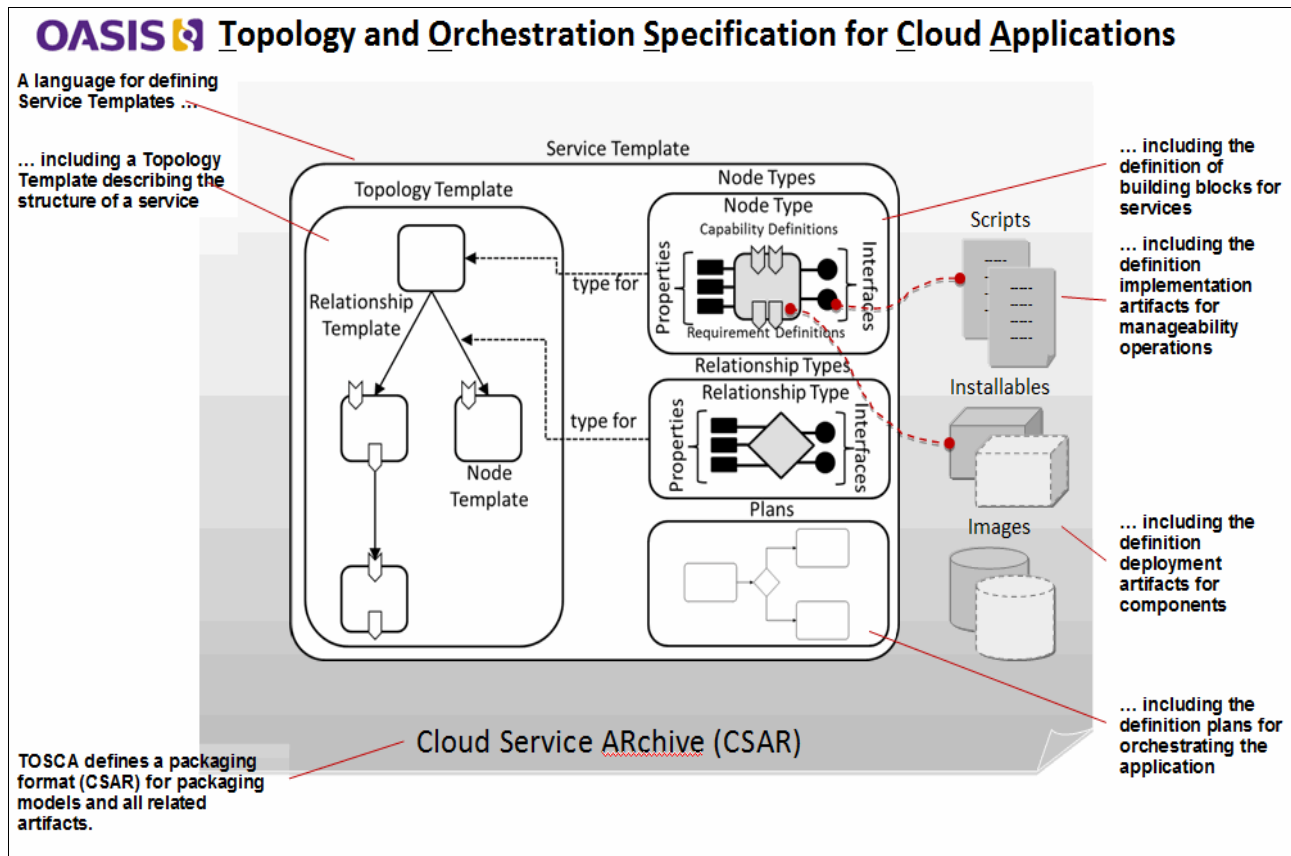


Figure 5-1 TOSCA model that defines a Cloud Service Archive (CSAR)

For more information about TOSCA, see the TOSCA portion of the OASIS website:

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=tosca](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca)

The new TOSCA standards are incorporated into all future releases of IBM cloud technologies. IBM SmartCloud Orchestrator already includes support for TOSCA-standard Cloud Service Archive (CSAR) files.

Support for the TOSCA standards is activated by installing and enabling the TOSCA Foundation Pattern Type. By default, this pattern type is installed in SmartCloud Orchestrator, but it is disabled.

To enable the TOSCA Foundation Pattern type in your environment, complete the following steps in the SmartCloud Orchestrator web-based GUI:

1. Click **Configuration** → **Pattern Types**.
2. Select **TOSCA Foundation Pattern Type**.
3. In the Status field, select **Enable**.

Figure 5-2 shows the Status field for the TOSCA pattern.


TOSCA Foundation Pattern Type	
<b>Description:</b>	TOSCA Foundation Components Pattern Type
<b>Status:</b>	 Unavailable [ <a href="#">Enable...</a> ]
<b>System Plug-ins:</b>	<a href="#">Show me all plug-ins in this pattern type</a>

Figure 5-2 Status field where pattern type is enabled

After the pattern type is enabled, you can import TOSCA-standard CSAR files that contain service templates. You can import the archives as virtual application patterns or virtual application templates.

For more information about enabling and the use of TOSCA pattern types in SmartCloud Orchestrator, see the SmartCloud Orchestrator 2.2 Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc\\_2.2/t\\_tosca\\_enabling.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc_2.2/t_tosca_enabling.html)

**Note:** The CSAR model as defined by TOSCA is similar to the model of the patterns in the current IBM cloud technology implementations.

The following standard package formats are used by IBM cloud technologies for patterns:

- ▶ Virtual system patterns: Python files (.py) and archive files (.tgz or .tar.gz) are used for these patterns, depending on the import and export methods.
- ▶ Virtual application and database patterns: Archive files (.zip) are used for these patterns.

## 5.2 Import and export prerequisites

This section describes the prerequisites for importing and exporting patterns. All of the cloud technologies that are described in this book support the use of the command-line interface (CLI) for moving patterns between them. Some of the patterns also support the REST API and the user interface GUI for moving patterns between technologies, as described throughout this chapter.

### 5.2.1 Hardware requirements

The only hardware requirement is using a device that can connect to the cloud technology with the appropriate network protocol. All three methods for importing and exporting patterns rely on HTTPS sessions. If your device loses connectivity to port 443 of any of your cloud systems, you cannot perform the functions that are described here.

### 5.2.2 Software requirements

The following software requirements must be met for importing and exporting patterns:

- ▶ A Windows or Linux operating system (needed to use the CLI)
- ▶ Java Runtime Environment (JRE) version 6 or higher
- ▶ A web browser with cookies enabled (required to run the web interface GUI)

In addition, the user who is running the processes must have the following permissions to export and import patterns:

- ▶ Access to create patterns
- ▶ Access to create catalog content
- ▶ Ownership of the pattern or read-only access to it

### 5.2.3 CLI download and setup

The CLI is used to administer and manage the cloud system in a scripted, non-graphical environment and includes the following features:

- ▶ Runs Python scripts and commands in a Jython scripting environment
- ▶ Uses REST APIs
- ▶ Can be started in interactive, command, or batch modes

Complete the following steps to download the CLI to your local system from any of the IBM cloud technology management consoles:

1. Go to the console's Welcome page, click **Download tooling**, and then click **Download command-line tool**. Figure 5-3 on page 149 shows this process in the IBM Workload Deployer console (other interfaces are similar).

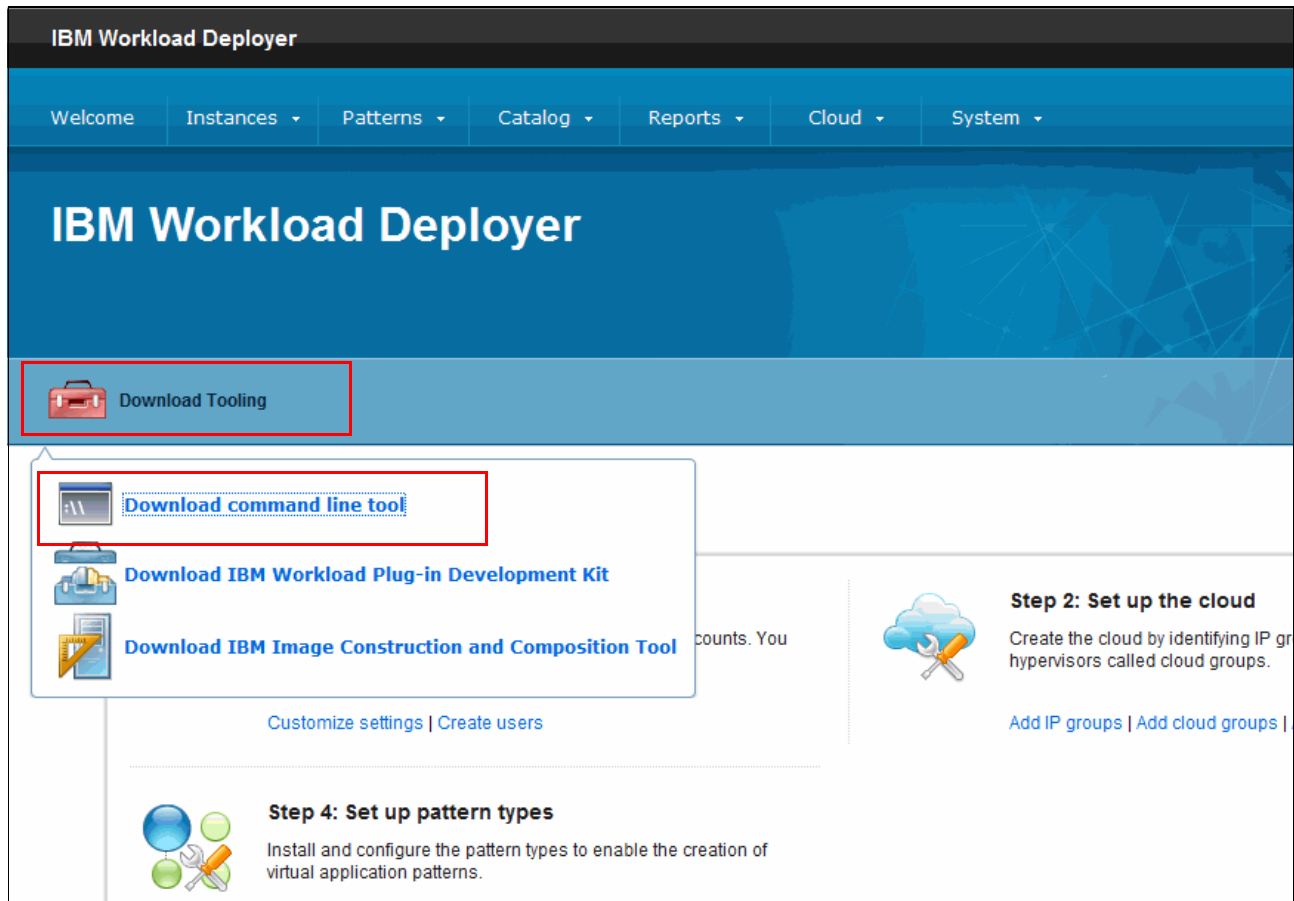


Figure 5-3 Downloading the CLI from the Workload Deployer console

2. Save the compressed archive file to your local hard disk drive and extract its contents. When extracted, a single directory is created with one of the following names:
  - `pure.cli` for IBM PureApplication System
  - `deployer.cli` for IBM Workload Deployer (IWD), IBM SmartCloud Orchestrator, IBM SmartCloud Provisioning, IBM Pattern Kit for Developers, and IBM SmartCloud Application Services (SCAS)
3. Ensure that the `JAVA_HOME` or the `PATH` environment variable is set in your local environment and that it points to the JRE location.

**Optional:** If you are using Windows Server 2003 or 2008, under the `<CLI_HOME>` directory (`deployer.cli` for example), create a registry file in the `lib/<version>` with the following line:

```
python.os-nt
```

## 5.2.4 Using the CLI

After the CLI is installed, you are now ready to use it to export and import patterns.

**Note:** Run commands from `<CLI_HOME>/deployer.cli/bin`, where `<CLI_HOME>` is the location of the extracted installation directory of CLI.

The CLI can be used in the following modes:

- ▶ Interactive mode: Run commands interactively while you are getting information from the cloud technology. This mode supports command history and a subset of Emacs commands with Control-key binding only from JLine.
- ▶ Command mode: Run a single command.
- ▶ Batch mode: Run a set of commands by using Python or Jython scripts.

Example 5-1 shows how to start the CLI to use each of the available modes. The remainder of this chapter describes specific pattern importing and exporting procedures that use each of these modes.

*Example 5-1 Commands for starting the CLI in the various modes*

---

Interactive mode

```
deployer -h <host> -u <user> -p <password>
```

Command mode

```
deployer -h <host> -u <user> -p <password> -c "<command>"
```

Batch mode

```
deployer -h <host> -u <user> -p <password> -f "<filepath>" <script arguments>
```

---

For many commands, the CLI uses the terms *resources*, *resource collections*, and *methods*. IBM cloud technologies manage different types of resources, such as script packages, patterns, virtual system instances, and environment profiles. Jython objects are used within the CLI to represent these resources and collections.

The terms feature the following definitions:

- ▶ Resource: An individual object representation that has both properties and methods.
- ▶ Resource collections: A grouping of one type of resource that has methods but does not have any properties.
- ▶ Methods: Operations that can be performed on a particular resource.

For more information about the CLI, see the IBM cloud technology product information centers. The information center for Workload Deployer is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/ct\\_usingcli.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/ct_usingcli.html)

## 5.3 Exporting and importing patterns

Pattern types can be exported and imported in different ways. In this section, we describe the processes that are used for performing these actions by using the built-in web GUI interface, the CLI, and by using sample scripts.

Some pattern types can also be managed with REST APIs. For more information about this approach, see the Information Center for the IBM cloud technology product you are using.

### 5.3.1 Working with virtual system patterns

Virtual system patterns (VSPs) can be exported and imported only by using the CLI, in which there are sample scripts that you can use to make the process easier.

#### Exporting and importing by using the CLI

In this section, we describe how to work with VSPs by using the CLI.

##### ***Using the patternToPython sample script to export a pattern***

The patternToPython sample script is in the `<CLI_HOME>/samples` directory and is used to export a pattern and the components that make up the pattern. Although the script does not download the images or script packages that are associated with the pattern, it does reference the images and packages in the exported file.

The output of the export is a generated Python script that has the following options:

- ▶ `-f <filename>` or `-filename <filename>`  
This option tells the script where to store the exported pattern. If not otherwise specified, the script is written to your standard output location.
- ▶ `-p <pattern>` or `-pattern <pattern>`  
This option specifies the name of the pattern to be exported. The name that you provide must uniquely identify the virtual system pattern to be exported. This option must be present when you are using the CLI in batch mode for this script to run properly.

The patternToPython script can be run by using two different command-line modes:

- ▶ **Interactive mode**  
In this mode, you specify only the name to be given to the exported file and select the virtual system pattern from a list.

For example:

```
deployer -h <hostname> -u <user> -p <password> -f samples/patternToPython.py  
-f <path>/MyExportedPattern.py
```

Upon completion, the pattern selection window within the CLI looks similar to what is shown in Example 5-2.

*Example 5-2 Pattern selection window*

---

```
1. MyPattern  
2. WebSphere cluster  
select a pattern to convert:
```

---

- ▶ **Batch mode**  
In this mode, you specify the name of the pattern to be exported and the location where the exported pattern is to be placed, as shown in the following example:  

```
deployer -h <hostname> -u <user> -p <password> -f samples/patternToPython.py -p  
"MyPattern" -f <path>/MyExportedPattern.py
```



Example 5-3 shows an example of the python code that you find in the .py file after the pattern is exported.

*Example 5-3 Sample output of the patternToPython script*

```
patternName = u'MyPattern'

patterns = deployer.patterns[patternName]
assert not patterns or patterns[0].name != patternName, _utos((u'A pattern
named "%s" is already defined.' % (patternName)))

virtualImages = [
    _findVirtualImage(u'WebSphere Application Server 8.5.0.0 32-bit RHEL 6
x86-64 (VMWare)', u'8.5.0.0', u'R2_mX_1222.02')
]

pattern = deployer.patterns << {
    "name": patternName
}

patternPart_1 = pattern.parts << _findPart(virtualImages[0], u'Custom nodes',
1)
```

To import the exported pattern into a cloud technology, run the following command by using the CLI that uses Batch mode:

```
deployer -h <hostname> -u <user> -p <password> -f <path>/MyExportedPattern.py
```

This command string imports the pattern into the new system as is if no other pattern in the system has the same name.

Figure 5-4 shows the topology for MyPattern. The pattern that is shown in Figure 5-4 uses VMWare (ESX), WebSphere Application Server 8.5, and a sample script package.

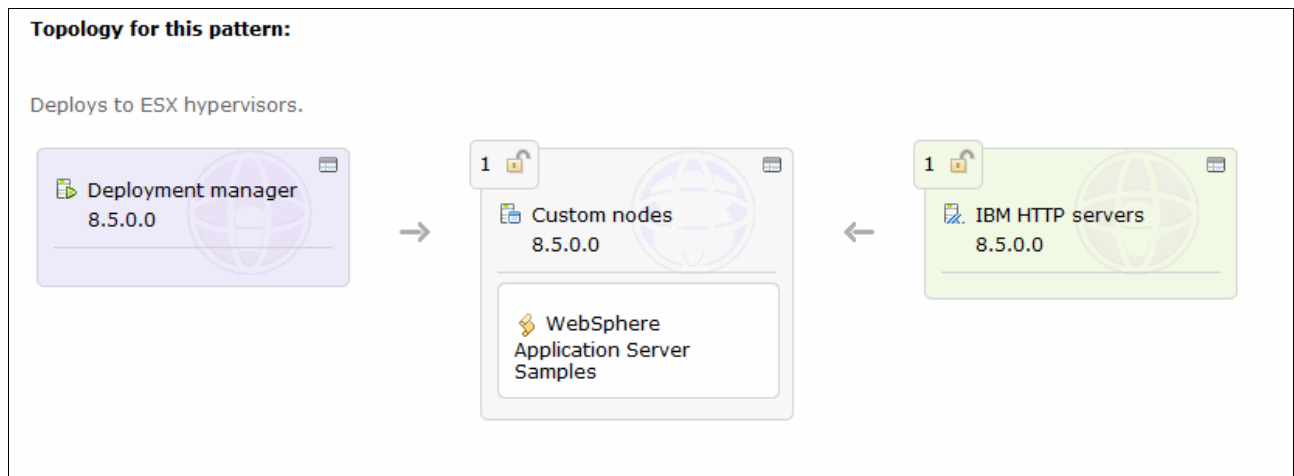


Figure 5-4 MyPattern topology as seen in the Patterns view in the web-based GUI

If the imported pattern references images that are not on the target cloud technology, the pattern import process fails. Example 5-4 on page 153 shows an example of the error message that is generated when a referenced image does not exist.

---

*Example 5-4 AssertionError when image referenced in imported pattern does not exist*

---

No virtual image named "WebSphere Application Server 8.5.0.0 32-bit RHEL 6 x86-64 (VMWare)" is defined.

---

To solve this type of referenced image problem, you can use a text editor to modify the exported Python script name that you used in the export process (MyExportedPattern.py) to use an image that is present on the target technology. To find the list of available images, you can use the CLI as shown in Example 5-5 in command mode.

---

*Example 5-5 Command-line sample for listing all images to a file*

---

```
deployer -h <hostname> -u <user> -p <password> -c deployer.virtualimages.list >>  
<path>/myImagesList.txt
```

---

You can also use the command line interactively and use the command in Example 5-5 to display in the CLI the various images available in the target technology.

If you cannot find an image that is already deployed on the target cloud technology, you can upload an image by following the instructions in the appropriate information center. For more information about the Workload Deployer, see this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r0m0/topic/com.ibm.worlodep.doc/pc/pc\\_r\\_cli\\_virt\\_image.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r0m0/topic/com.ibm.worlodep.doc/pc/pc_r_cli_virt_image.html)

Assuming the missing image was the only error that you encountered, your effort to import the pattern succeeds after the reference to the image is fixed. However, another error that is related to a missing script package can also occur. Example 5-6 shows an example of the error message that is generated when a pattern import fails because of a missing script package.

---

*Example 5-6 AssertionError when a needed script package (to import a pattern) does not exist*

---

Unable to find matching script for original pattern part "Custom nodes" (id 1), script "MyScriptPkg".

---

When this type of error is encountered, the image portions of the pattern definition are still imported, but the script package is not.

For more information about the patternToPython sample script, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r0m0/topic/com.ibm.worlodep.doc/dd/mp/mpt\\_imexpatt.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r0m0/topic/com.ibm.worlodep.doc/dd/mp/mpt_imexpatt.html)

### **Using the exportPattern sample script**

The exportPattern script is in the <CLI\_HOME>/samples directory. By default, this script captures the components of a pattern and the related script packages, but it downloads only the script packages. The script can also be used if you want to download all the components of a pattern by passing another option on the command line.

The output of the export is an archive file (.tgz or tar.gz). The exportPattern sample script has the following options for command parameters:

- -p <pattern\_name> or --pattern <pattern\_name>

This parameter specifies the name of the VSP to export. The name must uniquely identify the VSP. If you do not supply a name, a list of available VSPs is displayed so you can select one.

- `-t <target_directory>` or `--target <target_directory>`  
This parameter specifies the exported pattern components that are to be written to the target folder with the name of the archive file.
- `--passwords`  
This parameter includes BAS64 encoded passwords in the pattern export definition file. By default, all password properties are not exported.
- `--downloadAll`  
This parameter tells the script to download all of the associated artifacts of the pattern, including the virtual images, script packages, and any add-ons that are used.
- `-d | --download <filter_file>`  
This parameter indicates that only the artifacts that are listed in the filter file (JSON format) are downloaded.

The exportPattern script can be run (by using the options that were previously described) in the following command-line modes:

► **Interactive mode**

In this mode, you specify only the name of the file to export to and select the VSP from a list. The following example shows starting the script in Interactive mode:

```
deployer -h <hostname> -u <user> -p <password> -f samples/exportPattern.py -t
<path>/MyExportedPattern.tgz
```

Figure 5-2 on page 147 shows what a pattern selection window looks like.

► **Batch mode**

In this mode, you specify only the name of the file to export to and the location to place the export. The following example shows starting the script in Batch mode:

```
deployer -h <hostname> -u <user> -p <password> -f samples/exportPattern.py -p
“MyPattern” -t <path>/MyExportedPattern.tgz
```

After you export the pattern and extracted the archive file, the directory structure of the exported pattern looks similar to the structure that is shown in Figure 5-5.

Name	Size	Packed	Type	Modified
..			File folder	
add_ons			File folder	7/17/2013
images			File folder	7/17/2013
script_packages			File folder	7/17/2013
patterns.json	8,635	?	JSON File	7/17/2013

Figure 5-5 Directory structure of exported pattern

Regardless of which CLI mode you use, if you use the exportPattern script with a pattern that does not contain any script packages, the error that is shown in Example 5-7 appears.

*Example 5-7 Java IO error generated when a script package is not in the pattern to be exported*

---

```
java.io.IOException: java.io.IOException: This archives contains unclosed
entries.
```

---

Example 5-8 shows the JSON code that you find in the pattern's .json file from the extracted pattern archive file that was exported.

*Example 5-8 Sample output of the pattern's .json file*

---

```
{
  "advanced_options": [
  ],
  "name": "MyPattern",
  "parts": [
    {
      "add_ons": [
      ],
      "conceptualKey": "CustomNodeGroupPart",
      "concreteKey": "CustomNodePart",
      "configurations": [
      ],
      "count": 1,
      "name": "Custom nodes",
      "properties": [
        {
          "key": "numvcpus",
          "pclass": "HWAttributes",
          "value": "1"
        },
        {
          "key": "memsize",
          "pclass": "HWAttributes",
          "value": "2048"
        },
        {
          "key": "physcpures",
          "pclass": "HWAttributes",
          "value": "false"
        },
        {
          "key": "physmemres",
          "pclass": "HWAttributes",
          "value": "false"
        },
        {
          "key": "node_name",
          "pclass": "ConfigWAS",
          "value": "CloudBurstNode"
        }
      ],
    },
  ],
}
```

---

The exportPattern script with the downloadAll option is a good way to export everything within a pattern, including the definition, images, script packages, and add-ons. This can simplify the process of moving those components between different cloud technologies. The following example shows the command syntax for the downloadAll option with the exportPattern script, as used in CLI Batch mode:

```
deployer -h <hostname> -u <user> -p <password> -f samples/exportPattern.py -p
"MyPattern" -t <path>/MyExportedPattern.tgz --downloadAll
```

**Tip:** Exporting the virtual images by using the download option can take a long time, depending on the number and size of the images. You can reduce the process duration and the space that is required to export a pattern by updating the .json file to use an image that exists on the target cloud technology.

For more information about the exportPattern sample script, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/mpr\\_vsypsexim.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/mpr_vsypsexim.html)

### ***Using the importPattern sample script***

The importPattern script is in the <CLI\_HOME>/samples directory. By default, the script imports the components of a pattern and any related script packages that were exported through the exportPattern script in “Using the exportPattern sample script” on page 153. This script works only with archive files.

The script’s only source of input is a compressed archive file (.tgz or tar.gz). The script has the following mandatory option:

`-s <target_source_directory> or --source <target_source_directory>`

This option states that the pattern and its associated artifacts are to be uploaded from the named source file.

The importPattern script can be run only in Batch mode, where you specify the exact name of the pattern and its location of the exported pattern. The following example shows the use of the importPattern script in Batch mode:

```
deployer -h <hostname> -u <user> -p <password> -f samples/importPattern.py -s  
<path>/MyExportedPattern.tgz
```

If you are importing a pattern that does not contain all of the downloaded content in its archive file (presumably because of not using the downloadAll option in the exportPattern script), confirm that an image and the associated script packages and add-ons were placed on the target cloud technology before the import is done. The pattern import fails if any of the artifacts are not present.

For more information about the importPattern sample script, see the Workload Deployer Information Center page, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/mpr\\_vsypsimport.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/mpr_vsypsimport.html)

## **5.3.2 Working with Virtual Application Patterns**

Unlike VSPs, virtual application patterns (VAPs) can be exported and imported three different ways, as described in this section. The easiest method is by using the web-based GUI.

### **Exporting and importing by using the user interface**

The web-based GUI has built-in functionality with which you can export and import your VAP. The web-based GUI is best to use if the pattern to be exported is under 2 GB in size. If you attempt to use the GUI to export a pattern that is larger than that, a warning panel appears.

Complete the following steps to export a VAP by using the web-based GUI:

1. Click **Patterns** → **Virtual Applications**.
2. Click the down arrow for the type of virtual application.
3. Select the virtual application to export.
4. Click **Export**.

Figure 5-6 shows the Export button in the virtual application pattern window.

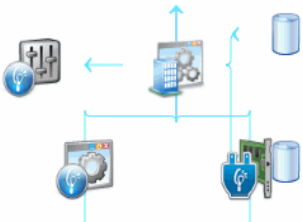

MyMobilePattern		Deploy	Open	Export
Application ID:	a-186738f2-ebdf-4df2-900f-406c077174f0			
Description:				
Created by:	cbadmin			
Updated by:	cbadmin			
Created on:	Jul 30, 2013 10:17:47 AM			
Updated on:	Jul 30, 2013 10:17:48 AM			
Preview:				
Access granted to:	Everyone [ read ] [ remove ] <input type="text" value="Add more..."/>			
Pattern type:	IBM Mobile Application Platform Pattern Type 6.0 (  Unlocked )			

Figure 5-6 Web-based GUI virtual application pattern export window

After the export process starts, a window opens in which you are prompted to confirm what you want to do with the exported pattern, as shown in Figure 5-7.

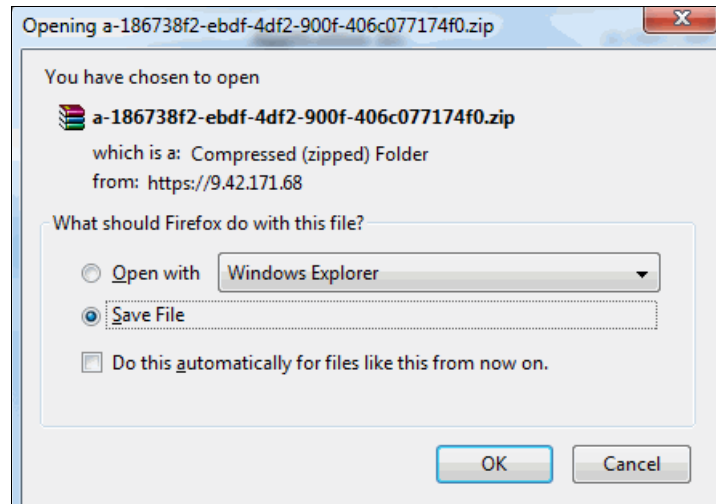


Figure 5-7 Exported virtual application window

If you choose to open the exported pattern to update or view it, you see the directory structure that is shown in Figure 5-8.

Name	Size	Packed	Type	Modified
..			File folder	
artifacts			File folder	7/30/2013
appmodel.json	5,181	1,002	JSON File	7/30/2013
appmodel_layout.json	1,193	223	JSON File	7/30/2013

Figure 5-8 Directory structure of exported virtual application pattern

The two JSON files show the contents of the exported virtual application pattern, as shown in Example 5-9.

Example 5-9 Content of appmodel.json sample file

```
{
  "layers": [
    {
      "nodes": [
        "myWorklight Server",
        "myWorklighDB",
        "Worklight Configuration",
        "Worklight Adapter",
        "Worklight Application",
        "myRPTDB"
      ],
      "id": "layer"
    }
  ],
  "model": {
    "nodes": [
      {
        "groups": {
        },
      }
    ]
  }
}
```



```

    "attributes": {
      "archive": "artifacts/1375000913334WorklightStarter.ear",
      "clientInactivityTimeout": 60,
      "propogatedOrBMTTranLifetimeTimeout": 300,
      "asyncResponseTimeout": 120,
      "totalTranLifetimeTimeout": 120,
      "ignoreFailedIfix": true
    },
    "type": "EAR",
    "id": "myWorklight Server"
  },

```

---

Importing a VAP follows a process that is similar to exporting a VAP. To import a VAP, complete the following steps in the web-based GUI:

1. Click **Patterns** → **Virtual Applications**.
2. Click the **Import** icon, as shown in Figure 5-9.

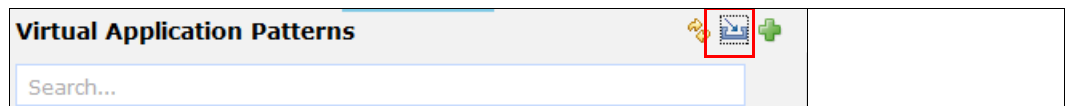


Figure 5-9 Import icon that is used to import VAPs

3. Click **Browse** and select the wanted file to import, as shown in Figure 5-10.

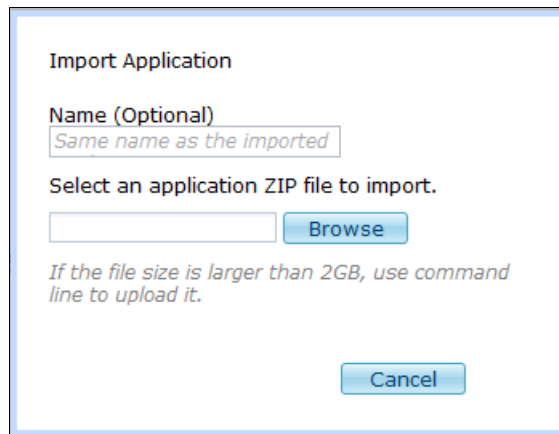


Figure 5-10 Selecting a VAP to import

The upload begins when you select the file to import. The process status is displayed near the top of the web-based GUI and shows any errors that might occur.

## Exporting and importing by using the REST API

As with the web-based GUI, the REST API can be used to export and import virtual application patterns. Some of the necessary commands can be run within the browser, while others require a separate REST API client.

Complete the following steps export a virtual application pattern by using the REST API:

1. In your browser or REST API client, browse to the following location:  
[https://<hostname>/resources/applicationPatterns/<app\\_id>?zip](https://<hostname>/resources/applicationPatterns/<app_id>?zip)

As shown in Example 5-10, you can find your app\_id by using the REST API by using GET /resources/applicationPatterns/. The application ID has a URL in front of it. You need to capture the ID only, as shown in bold in Example 5-10.

*Example 5-10 JSON list of a virtual application pattern*

---

```
{
  "last_modifier": "cbadmin",
  "content_type": "application/json",
  "app_storehouse_base_url": "a-186738f2-ebdf-4df2-900f-406c077174f0/",
  "app_name": "MyMobilePattern",
  "patterntype": "worklight.ptype",
  "locked": "false",
  "creator": "cbadmin",
  "create_time": "2013-07-30T14:17:47Z",
  "last_modified": "2013-07-30T14:17:48Z",
  "access_rights": {
    "cbadmin": "F",
    "_group_": "Everyone": "R"
  }
}
```

---

Based on the example, after you locate the application ID, the address in your browser look like the following example:

`https://<hostname>/resources/applicationPatterns/a-186738f2-ebdf-4df2-900f-406c077174f0?zip`

2. If you are not already logged in to the cloud technology's web-based GUI, you are prompted to provide your user name and password.
3. To complete the export process, you are presented with a window that is similar to the one that is shown in Figure 5-7 on page 158.

Importing a VAP archive with the REST API is similar to exporting an archive, but requires a REST client. Complete the following steps:

1. In your REST API client, use the following syntax:  
POST <url>/resources/applicationPatterns
2. Set your headers to include the following information:
  - Content-Type: application/zip
  - Content-Type: X-IBM-Workload-Deployer-API-Version: 3.1
3. Enter the command. This example uses cURL, which is found in most UNIX operating systems, but any REST client can be used. The following syntax is used:

```
curl --basic -u <user>:<password> -H 'Content-Type: application/jzip' -H  
'X-IBM-Workload-Deployer-API-Version: 3.1' -k https://<url of cloud  
technology>/resources/applicationPatterns/ --data-binary "@<filename.zip>"
```

The response should look similar to what is shown in Example 5-11 on page 161.

```
{
  "content_type": "application/json",
  "last_modifier": "cbadmin",
  "create_time": "2013-08-01T12:42:25Z",
  "last_modified": "2013-08-01T12:42:26Z",
  "access_rights": {
    "cbadmin": "F"
  },
  "content_md5": "335959D325D92391610A52C01EC29C09",
  "app_type": "application",
  "app_name": "MyWebApplication",
  "app_id": "a-483ec5d0-c705-443e-8d8e-ad1742a9ff00",
  "locked": "false",
  "creator": "cbadmin",
  "Collection": "."
}
```

---

For more information about the use of the REST API for importing and exporting VAPs, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc\\_restapiartifacts.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc_restapiartifacts.html)

## Exporting and importing by using the CLI

In addition to the web-based GUI and the REST API, you can export and import VAPs by using the CLI. The use of the CLI is preferred when the pattern exceeds 2 GB in size.

### Exporting

The export process can be performed in the CLI by using the following Interactive or Command modes:

- Interactive

By using this mode, you can export a VAP that is based on its name or index number. The easiest approach often is to use the name method.

To download the pattern that is based on its name, use the following command:

```
deployer.applications.list({"app_name": "MyPattern"})[0].download("<path>/filename.zip")
```

To download the pattern using the index number, use the following command:

```
deployer.applications[0].download("<path>/filename.zip")
```

- Command mode

Exporting a pattern by using Command mode is similar to exporting the pattern in Interactive mode. The syntax is shown in the following example:

```
deployer -h <hostname> -u <user> -p <password> -c
"deployer.applications[0].download('<path>filename.zip')"
```

**Note:** The main difference between the syntax in Interactive and Command modes is the use of a single quotation mark in the command path instead of double quotation marks.

### **Importing**

The command syntax is the same for importing a VAP pattern as it was for exporting a pattern, and the process can be done in Interactive or Command mode. Interactive mode uses the following syntax:

```
>>> deployer.applications.create("<path>/filename.zip")
```

For more information about the use of the CLI for importing and exporting VAPs, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/air\\_apppattern.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/air_apppattern.html)

## **5.3.3 Working with database patterns**

Exporting and importing database patterns follows a process that is similar to the process that is used for VAPs, except that the web-based GUI is not supported for porting database patterns.

### **Exporting and importing by using the REST API**

The REST API can be used to export and import database patterns. Some of the commands can be run within the browser, while others require a separate REST API client.

#### **Exporting**

Complete the following steps to export a database pattern by using the REST API:

1. In your browser or REST API client, browse to the following location:

```
https://<hostname>/resources/applicationPatterns/<app_id>?zip
```

As shown in Example 5-10 on page 160, you can use the REST API to find the app\_id by running the following command string:

```
GET /resources/applicationPatterns/
```

As shown in this example, the application ID is preceded by a URL. You need to capture only the ID, which is shown in bold in Example 5-10 on page 160. When complete, the address in your browser looks similar to the following example:

```
https://<hostname>/resources/applicationPatterns/a-29b31ddb-31a0-4f0c-a1bc-53177a0c0093?zip
```

2. If you are not already logged in to the cloud technology's web-based GUI, you are prompted for your user name and password. Enter those credentials.
3. To complete the export process, a window opens that is similar to the one that is shown in Figure 5-7 on page 158.

#### **Importing**

The process of importing a database pattern by using the REST API is similar to exporting a pattern, but requires a separate REST client. The steps are identical to the steps that are used to import a VAP. In the following steps, the difference in using the extracted JSON file instead of the full archive is shown.

Complete the following steps:

1. Enter the following syntax in your REST API browser client:

```
POST <url>/resources/applicationPatterns
```

2. Set your headers to include the following information:
  - Content-Type: application/json
  - Content-Type: X-IBM-Workload-Deployer-API-Version: 3.1
  - Accept: application/json
3. Extract the contents of your archive file and open your appmodel.json file.
4. Copy and paste the contents of the appmodel.json file into the body of the request and click **Send**.
5. The response back should look similar to what is shown in Example 5-12.

---

*Example 5-12 REST API client response*

---

```
1. Status Code: 201 Created
2. Cache-Control: must-revalidate, max-age=0, private, no-cache
3. Content-Encoding: gzip
4. Content-Length: 271
5. Content-Type: application/json
6. Date: Wed, 31 Jul 2013 18:39:25 GMT
7. Location:
https://9.42.170.108:443/resources/applicationPatterns/a-04a14abe-e2bb-49f5-8aa
0-987793dc3775
8. Server: IBM WebSphere sMash/1.1.1.6.31742
```

---

If you do not have a browser-based REST client, you can also use cURL, which is built into most UNIX operating systems. With this approach, you can paste the contents of the appmodel.json file into the data stream or pass the file. The following syntax is used for passing the file:

```
curl --basic -u <user>:<password> -H 'Content-Type: application/json' -H
'X-IBM-Workload-Deployer-API-Version: 3.1' -k https://<url of cloud
technology>/resources/applicationPatterns/ --data "@<filename.json>"
```

The response back should look similar to what is shown in Example 5-13.

---

*Example 5-13 JSON Response for importing a database pattern using cURL*

---

```
{
  "content_type": "application/json",
  "last_modifier": "cbadmin",
  "create_time": "2013-07-31T19:42:06Z",
  "last_modified": "2013-07-31T19:42:06Z",
  "access_rights": {
    "cbadmin": "F"
  },
  "content_md5": "1344A3A35B911A92F7CA68D1B848AD39",
  "app_type": "database",
  "app_name": "MyDBPattern",
  "app_id": "a-4bf87846-c2b3-4d08-86f2-744a87bada6e",
  "locked": "false",
  "creator": "cbadmin",
  "Collection": "."
}
```

---

For more information about the use of the patterns, see the section about VAPs in the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc\\_restapiartifacts.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc_restapiartifacts.html)

### 5.3.4 Working with pattern types

Most VAPs require a pattern type. For more information about pattern types, see 2.3.1, “Elements” on page 33.

Pattern types can be imported from one cloud technology into another cloud technology, but pattern types cannot be exported. Pattern types can be imported by using the web-based GUI, the CLI, or REST API. The web-based GUI should be used only if the pattern to be imported is less than 2 GB.

Complete the following steps to import a pattern type by using the web-based GUI:

1. Click **Cloud** → **Pattern Types**.
2. Click the green plus sign (+) on the toolbar. This plus sign is called the New icon, as shown in Figure 5-11.

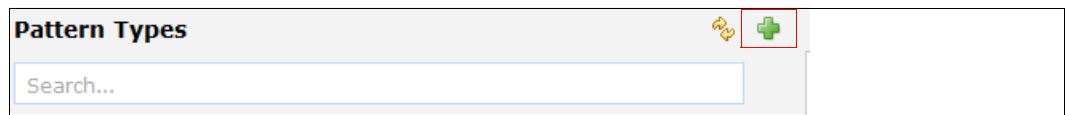


Figure 5-11 Importing a pattern type by using the web-based GUI

3. Provide the following details as necessary:
  - If you are uploading a local file, go to the Local tab and click **Browse** to select the .tgz file that contains the pattern type.
  - If you are uploading a remote file, go to the Remote tab and then specify the URL of the file. If prompted, enter a user name and password,

Figure 5-12 on page 165 shows the window in which you select a pattern type to import.

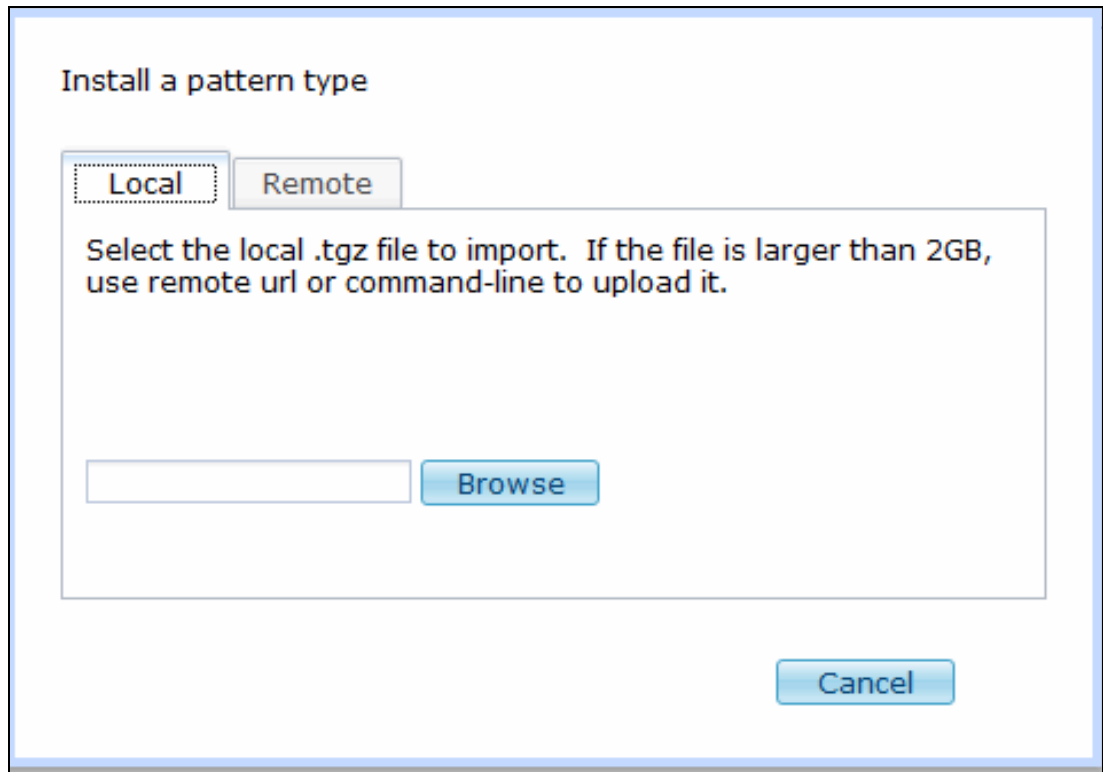


Figure 5-12 Selecting pattern type to import

4. Click **OK**.

To use the imported pattern type, you must accept any license agreements that are presented, configure any needed plug-ins, and enable the pattern type for use.

For more information about the use of the web-based GUI for importing pattern types, see the Workload Deployer Information Center page on pattern types, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apt\\_importpatterntype.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apt_importpatterntype.html)

The information center also has more information about importing pattern types by using the following alternative approaches:

► CLI:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt\\_app\\_patterns/cli/cliForWorkloadConsole.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt_app_patterns/cli/cliForWorkloadConsole.html)

► REST API:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc\\_restapipatterntype.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/apc_restapipatterntype.html)



### 5.3.5 Working with database workload standards

Database workload standards enable cloud administrators to provision databases with a standardized deployment approach. Workload standards are sets of configuration settings that are used to create databases, tune instance configurations, load data, and so on. By default, IBM cloud technologies include the following predefined database workload standards that are based on common requirements:

- ▶ Departmental Transaction
- ▶ Data Mart

Customized database workload standards provide the capability to create and tune databases to meet your specific requirements or corporate standards by using the two workload standards.

Database workload standards can be imported and exported by using the CLI and web-based GUI.

#### Exporting by using the CLI

The CLI can be used in Interactive or Command mode to export database workload standards.

The easiest way to export a workload standard by using the CLI is in Interactive mode by using the database workload standard ID. You can find the ID by running the following command in the CLI:

```
>>> deployer.dbworkloads.list
```

The output of the command is shown in Example 5-14. The parts that you must capture are shown in bold.

*Example 5-14 Output of list command*

---

```
{
  "description": "",
  "id": "7ccb5cd2-7b1f-48cd-ae8f-a2f0dc0965bd",
  "initial_disk_size": "1",
  "is_system": "false",
  "name": "WL_DB",
  "rate": "3",
  "version": "1.1.0.6",
  "workload_file": "WLRTDB.zip",
  "workload_type": "Departmental OLTP",
  "workloadfiles": (nested object)
}
```

---

The following syntax is used to download the database workload standards in Interactive mode:

```
deployer.dbworkloads.get("<dbworkloadID>").workloadfiles.download("<workload_file>
", "<local_path>")
```

For more information about the use of the CLI in relation to database workload standards, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt\\_app\\_patterns/cli/cliForWorkloadConsole.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt_app_patterns/cli/cliForWorkloadConsole.html)

## Importing by using the web-based GUI

The easiest way to import a database workload standard pattern is by using the web-based GUI. Complete the following steps to import or create a database workload standard:

1. Click **Catalog** → **Database Workload Standards**.
2. On the Database Workload Standards page, click the green plus sign (+), as shown in Figure 5-13.

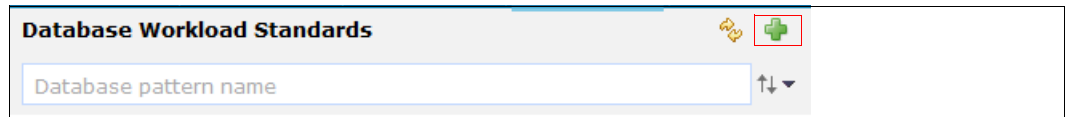


Figure 5-13 Importing a database workload standard

3. In the Database Workload Standards window, complete the following steps:
  - a. Enter values for the following fields:
    - **Name:** A unique name to identify the database workload standard.
    - **Description:** Identifying details to further describe the standard, if needed.
    - **Workload type**  
Initial disk size specifies the available space when the database is first deployed. It must be a number 0 - 500.  
Storage multiplier is used to determine the user data size. The default value is recommended. The range that is permitted is 1.0 - 3.0.
  - b. Click **Browse** or **Upload file** and browse to the location of the archive file that contains the workload standard scripts. The archive file must have a .zip extension and cannot begin with a number or the underscore (\_) character. The file name cannot be more than 18 characters and can contain only alphanumeric characters.
  - c. Click **Save**.

Figure 5-14 shows the Database Workload Standards window.

The screenshot shows a web-based GUI window titled "Database Workload Standards". Inside the window, there is a heading "Specify options for your customized database workload standard." followed by a form. The form contains the following fields and controls:

- Name:** A text input field containing the text "untitled".
- Description:** An empty text input field.
- Workload type:** A dropdown menu with "Departmental Transactional" selected.
- Initial disk size (GB):** A text input field containing the value "1".
- Storage multiplier (User data to physical disk ratio):** A text input field containing the value "3".
- Upload file (.zip):** A text input field followed by a "Browse" button.

At the bottom right of the form area, there are two buttons: "Save" and "Cancel".

Figure 5-14 Importing a database workload standard by using the web-based GUI

You can now view both the default and customized database workload standards in the list that is displayed.

When you deploy a database from a customized database workload standard, do not specify a database compatibility mode or upload a schema file.

For more information about the use of the web-based GUI for importing database workload standards, see the Workload Deployer Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt\\_app\\_patterns/db/dbt\\_customwls.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/virt_app_patterns/db/dbt_customwls.html)

## 5.4 Portability example that uses IBM SCAS and PureApplication System

This section provides detailed information about how to export and import a VAP from IBM SCAS to IBM PureApplication System. The example that is provided uses the following components:

- ▶ IBM Mobile Application Platform Type 6.0, which is dependent upon the following pattern types:
  - Foundation Pattern type 2.0.0.6
  - Web Application Pattern type 2.0.0.5
  - IBM OS image for Red Hat Linux Systems 2.0.0.3
- ▶ Two database workload standards
- ▶ Worklight Starter Virtual Application Pattern

For more information about designing a virtual application pattern in SCAS, see 2.2.4, “Design example” on page 26.

### 5.4.1 Exporting from SCAS

Complete the following steps to export your mobile application from SCAS by using the web-based GUI:

1. Click **Patterns** → **Virtual Applications**.
2. Click the down arrow and go to **IBM Mobile Application Platform type 6.0**.
3. Select your pattern name, in this case **WorklightStarter**. The contents of this pattern can be found in 2.2.4, “Design example” on page 26. The pattern topology is shown in Figure 5-6 on page 157.
4. Click **Export** and then use the window to save your pattern archive, as shown in Figure 5-7 on page 158.

### 5.4.2 Preparing PureApplication System to import the VAP

Before the VAP is imported into PureApplication System for deployment and testing, complete the following steps:

1. Log in to the PureApplication System Workload Console by using the web-based GUI.
2. Browse to **Cloud** → **Pattern Types**.
3. Verify that the following pattern types are installed:
  - Foundation Pattern type 2.0.0.6
  - Web Application Pattern type 2.0.0.5
  - IBM Mobile Application Platform Type 6.0

Figure 5-15 on page 170 shows an example of the listing for the Foundation Pattern type installed and enabled.




Foundation Pattern Type 2.0	
Foundation Pattern Type 2.0.0.2	
Foundation Pattern Type 2.0.0.6	

Figure 5-15 Listing for the Foundation Pattern type

Figure 5-16 shows that the Mobile Pattern Type is installed and enabled.



IBM Mobile Application Platform Pattern Type 6.0	
IBM Mobile Application Platform Pattern Type 6.0.0.0	

Figure 5-16 Listing for the Mobile Pattern type

4. If the pattern types are not installed, install them by using the following syntax in the CLI:  
`deployer.patterntypes.create("<path>/<filename>")`  
The file name for the mobile pattern type is: `worklight.ptype-6.0.0.0.tgz`.
5. For the pattern type to work properly, two other files (WLRPTDB.zip and WLRTDB.zip) are included with it and can be installed by using the same syntax that was provided in step 4.
6. After the pattern types are confirmed or installed, be sure to enable them as shown in Figure 5-2 on page 147.
7. Confirm that a version of an OS image is available by clicking **Catalog** → **Virtual Images**.
8. Confirm that the licenses for the OS image were accepted and that the image is set to deploy by default for virtual applications by clicking **Cloud** → **Default Deploy Settings**.

Figure 5-17 shows the default OS image to be used for VMWare virtual applications.


### Default Deploy Settings

Define the default virtual image for deploying shared services and virtual applications

NOTE: Only supports 64-bit hypervisors and images.

**Hypervisor Type:** **ESX**

Set the default image below:

Name	License Agreement	Version	Description	Reference ID
IBM OS Image for Red Hat Linux Systems	 Accepted	2.0.0.3	IBM OS Image for Red Hat Linux Systems	136

[Change](#)

Figure 5-17 Virtual applications default deploy settings

For more information about importing the mobile pattern type into IBM PureApplication System, see the IBM Worklight Information Center, which is available at this website:

[http://pic.dhe.ibm.com/infocenter/wrklght/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c\\_pureapp\\_installation.html](http://pic.dhe.ibm.com/infocenter/wrklght/v6r0m0/topic/com.ibm.worklight.help.doc/pureapp/c_pureapp_installation.html)

### 5.4.3 Importing a VAP into IBM PureApplication System

To import a VAP into PureApplication System, complete the following steps in the PureApplication System web-based GUI:

1. Click **Patterns** → **Virtual Applications**.
2. Select the down arrow and go to **IBM Mobile Application Pattern Type**.
3. Click **Import**.
4. Click **Browse** and then find your VAP that is named `WorklightStarter.zip`. The import starts when file is selected (see Figure 5-9 on page 159 and Figure 5-10 on page 159 for examples).

The pattern is imported successfully if all of the requirements were met (for more information, see 5.4.2, “Preparing PureApplication System to import the VAP” on page 169). However, this particular IBM Mobile Application Pattern is unlikely to deploy successfully unless it is updated because some identification issues within the `appmodel.json` file exist that are related to pointing to specific database workload standards.

If this occurs, you see an error message (`java.lang.NullPointerException`) when you attempt to deploy the pattern. To overcome this error, complete the following steps:

1. Log in to your target cloud technology by using the CLI and run the following command:  
`deployer.dbworkloads.get`
2. Capture the two IDs for the `WL_DB` and `WL_RPTDB` database workload standards, as shown in Example 5-15.

*Example 5-15 Highlighted WL\_DB id*

---

```
{
  "description": "",
  "id": "7ccb5cd2-7b1f-48cd-ae8f-a2f0dc0965bd",
  "initial_disk_size": "1",
  "is_system": "false",
  "name": "WL_DB",
  "rate": "3",
  "version": "1.1.0.6",
  "workload_file": "WLRTDB.zip",
  "workload_type": "Departmental OLTP",
  "workloadfiles": (nested object)
},
```

---

3. Extract the pattern archive file.
4. Go to the directory of the extracted pattern archive file and open the `appmodel.json` file.
5. Update the IDs of the database workload standards to what was captured in step 4.
6. Save the file.
7. Build a compressed file that contains the new file and all of the other artifacts and import the pattern again.

When all of the steps are completed, you can deploy and test your pattern to verify that it works.

**Tip:** This error and fix is specific to the Mobile Worklight Pattern, but might exist in other VAPs. A leading practice is to check the source of the VAP in the `appmodel.json` file or the Source tab in the Virtual Application Builder web-based GUI.

#### 5.4.4 Deploying your pattern

Complete the following steps in the PureApplication System web-based GUI to deploy your newly imported VAP:

1. Click **Patterns** → **Virtual Applications**.
2. Select the down arrow and go to **IBM Mobile Application Pattern Type**.
3. Select your VAP, in this case, **Worklight Starter**.
4. Select **Deploy**, as shown in Figure 5-18.

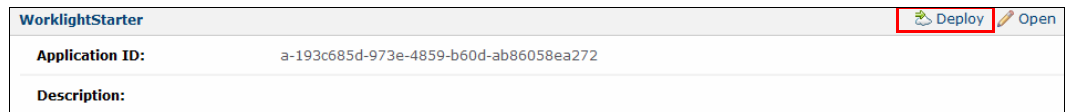


Figure 5-18 Deploying VAP Mobile pattern

5. Enter the information as needed for your deployment properties, as shown in Figure 5-19.

Figure 5-19 Mobile VAP Deployment properties



6. Check the status of the deployment by selecting **clicking here**, as shown in Figure 5-20.

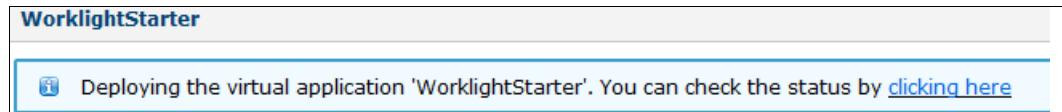


Figure 5-20 VAP status

Clicking the hyperlink takes you to the Virtual Application Instances page within the web-based GUI. The example in Figure 5-21 shows the progress of the VAP deployment. When the VAP is fully deployed, a green Start icon appears next to its listing.

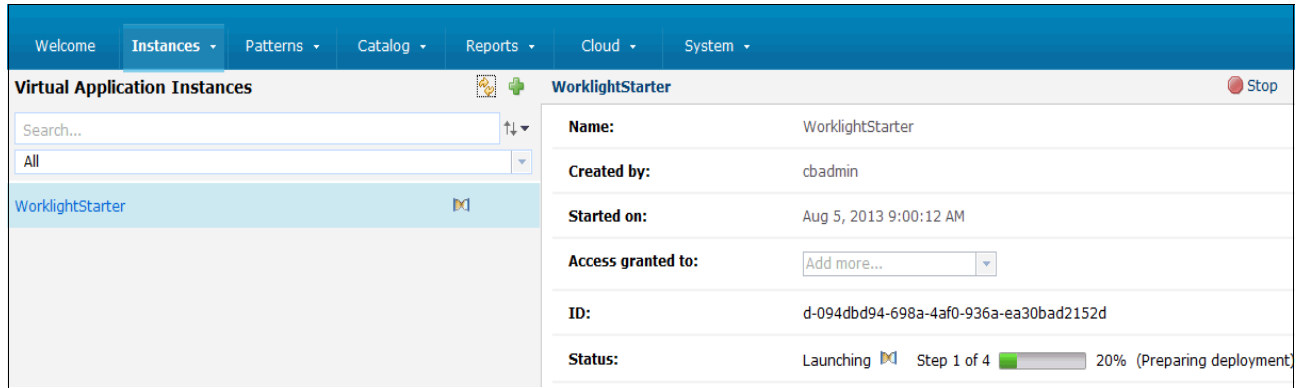


Figure 5-21 Progress of VAP deployment

7. Open your deployed pattern and click **WebSphere Application Server endpoint**, as shown in Figure 5-22.

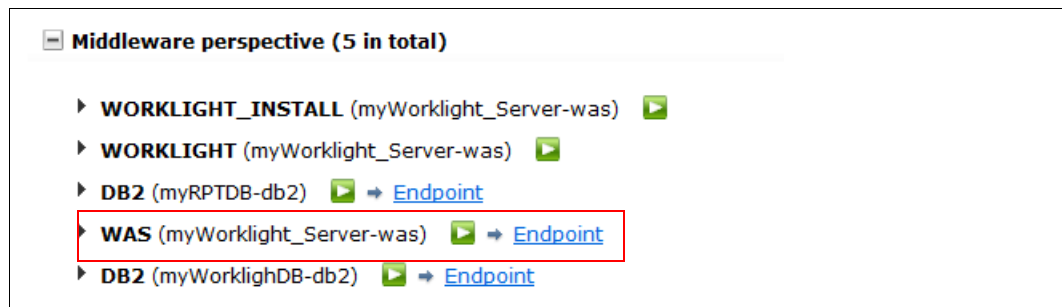


Figure 5-22 Worklight Starter deployed VAP

If everything works correctly, the application looks like in a web browser what is shown in Figure 5-23 on page 174. You successfully ported a virtual application pattern, database workload standards, and an application from an externally provided cloud technology (SCAS) to an internal private cloud technology (PureApplication System).

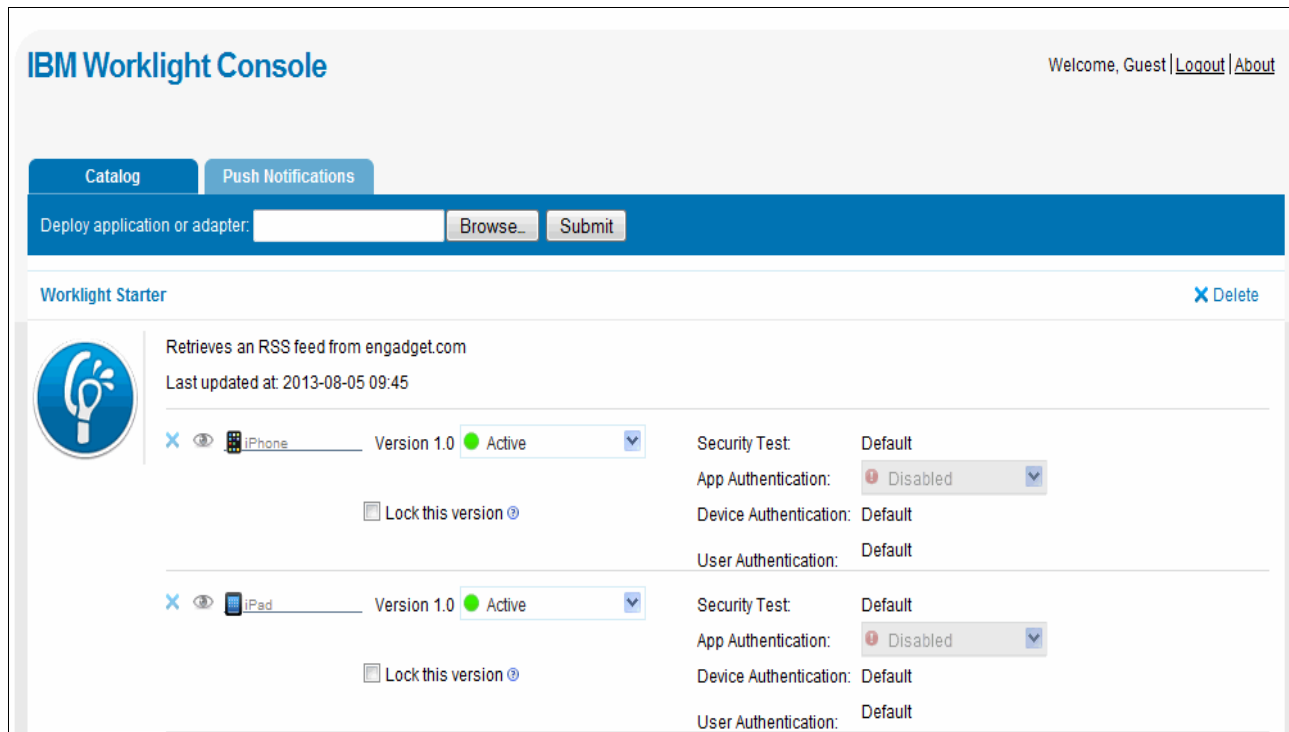


Figure 5-23 Worklight Starter console

## 5.5 Leading practices

When patterns are ported, several important configuration decisions must be made. The leading practices that are described in this section are designed to help.

### 5.5.1 Uniqueness

Patterns are configured with multiple components, all of which require the following items to be defined to ensure portability:

- **Name:** Make sure that your pattern and script package names are unique. By contrast, image names do not require uniqueness. The system includes built-in features to prevent the same names from being used.

Figure 5-24 shows the message that is displayed if you try to give a script package or pattern a conflicting name.

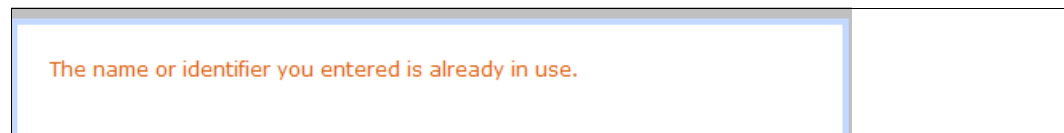


Figure 5-24 Naming error message

- **Version:** If the image name is the same, make sure that the component can be uniquely identified with version. Figure 5-25 on page 175 shows two images with the same name.



IBM OS Image for Red Hat Linux Systems	
IBM OS Image for Red Hat Linux Systems	

Figure 5-25 Image with the same name

If you select both of these images and dig deeper, you find one is at version 2.0.0.1, and the other at version 2.0.0.3. It is important to ensure that you select the correct image when you are porting or designing your pattern. In some cases, patterns can be changed with a lower or higher number version of the image within the pattern designer.

Take note of the various names and identifiers of pattern components so they can be edited if necessary before they are imported into another system. An IBM technote with details about moving a VSP from Workload Deployer to PureApplication System, including a description of the issues that relate to names and version numbers, is available at this website:

<http://www-01.ibm.com/support/docview.wss?uid=swg21609202>

## 5.5.2 Size Limits

When patterns that are larger than 2 GB are imported and exported, the CLI or REST API must be used. The web-based GUI can be used only when the file to be imported or exported is under the 2 GB size limit. If you attempt to work with a larger file in the web-based GUI, a warning window opens that is similar to the one that is shown in Figure 5-10 on page 159.

## 5.5.3 Security

For security purposes, it is recommended that you not pass the password when the CLI is started and used. An alternative method is to use a tool within the CLI to encode the password so it is not passed in human-readable form. Complete the following steps:

1. Run the **savePassword** script from within the <CLI\_HOME>/bin directory, as shown in the following example:

```
savePassword <filename> <password>
```

**Note:** Do not include any path or directory information in the command for <filename>. On Windows operating systems, the password file is saved to the folder that corresponds to %APPDATA%. On Linux or UNIX operating systems, the password file is saved to the directory that corresponds to \$HOME.

2. Use the **--passfile** option in the command instead of **-p**, as shown in the following example:

```
deployer -h <hostname> -u user --passfile <filename>
```

By using this recommended security action, you prevent the password from being displayed or sent in clear text. This technique can be useful when you are automating your CLI scripts or commands.

### 5.5.4 Automation

If your enterprise must automate the operations of the cloud technologies that you are using, it is recommended that you use the CLI or REST API. The CLI must be used for virtual system patterns. If VSPs are a requirement, make sure that your automation engine can support calling a shell script, such as the CLI, or has access to a REST client.



## Case studies

This chapter examines an end-to-end sample scenario of pattern adoption, which is framed in the context of Insurance Company X. This fictitious company is looking for a way to speed up its deployment of mobile applications to support car insurance claims from customers.

The chapter shows how Insurance Company X can use patterns to get started quickly in a public cloud setting and support rapid application development and deployment. When company leaders experience the value of patterns, they decide to bring their mobile application infrastructure in-house and deploy their mobile enterprise architecture as a private cloud solution.

The chapter includes the following topics:

- ▶ Enabling insurance on the go by using the IBM Mobile Platform Pattern
- ▶ Using patterns to cloud-enable a mobile application in the public cloud
- ▶ Transitioning to a private cloud for added flexibility and control

## 6.1 Enabling insurance on the go by using the IBM Mobile Platform Pattern

Insurance Company X wants to provide a mobile application to its customers to manage their automobile insurance, including updating account information, making payments, and filing claims. The company wants to build, deploy, and test its application quickly by using a flexible, virtualized infrastructure with which it can respond to the rapidly changing requirements and technologies in the mobile space.

### 6.1.1 Background

To develop and run its mobile application, Insurance Company X chose to use IBM Worklight. IBM Worklight helps organizations extend their business to various mobile devices (including smartphones and tablets from different providers) by providing open, standards-based tools to develop, run, and manage mobile applications.

Insurance Company X wants to ensure that its application runs easily on Android and iOS devices without the need for much native coding. In this regard, IBM Worklight simplifies cross-platform development and speeds each new application's time-to-market. In addition, IBM Worklight is available for cloud-based deployments in the IBM Mobile Platform Pattern, which is an IBM pattern of expertise that provides integrated middleware components, deployment capabilities, and lifecycle management tools.

Insurance Company X wants to use a virtualized cloud infrastructure for its mobile application to support workload consolidation and reduce hardware costs. The company does not have significant hardware and infrastructure management skills in-house, so it plans to begin by adopting a public cloud-based mobile solution. This allows the company to get started quickly, with minimal investment, to prove whether the IBM Worklight mobile solution can meet its needs.

After proving the feasibility of the solution, Insurance Company X wants to have an easy method for transitioning its application content from the company's development and test environments to its production servers, and to get its mobile application updates to market as quickly as possible. IBM patterns of expertise, such as the Mobile Platform Pattern, provide workload compatibility across many infrastructure environments, which makes it easy to move custom patterns from one environment to another. Patterns also provide rapid deployment features that help get applications fully deployed and configured in minutes.

### 6.1.2 Solution requirements

For application development and testing, Insurance Company X's mobile cloud solution must provide the following capabilities:

- ▶ A development workbench that integrates with the public cloud
- ▶ Automatic test environment definition on the cloud provider, including all server and database components
- ▶ Easy deployment of the test environment on the cloud provider
- ▶ Simple tools to develop shared application assets that run on Android and iOS
- ▶ Rapid provisioning of new test servers to the cloud (under 60 minutes)

- ▶ Quick application updates to pre-provisioned servers in the cloud (under 30 seconds) to enable easy testing of code changes
- ▶ An ability to quickly package the application pattern for production deployments

For production, the solution requirements still include capabilities that are related to rapid provisioning and application updates. The solution also must provide robust qualities of service, including clustering, failover, elasticity, and the ability to update the application and middleware without requiring an outage.

The following key service-level requirements for the production environment are mandated by the company:

- ▶ The application must run in a clustered configuration, with a minimum of two Worklight servers running always.
- ▶ If there is Worklight server failure, the cloud provider must automatically detect the failure and add a new server to the cluster. The new server must be available within 15 minutes of the initial failure.
- ▶ If CPU usage across the cluster exceeds 80% for more than 2 minutes, the cloud provider must add extra server capacity to bring usage below the threshold.
- ▶ If CPU usage across the cluster drops below 20% for more than 2 minutes, the cloud provider must remove servers from the cluster and return the infrastructure resources to the shared cloud pool.
- ▶ HTTP session caching must be provided so that, if there is a server failure, the environment still maintains session affinity for active requests.
- ▶ Basic monitoring for the virtual hardware and middleware must be available to determine the overall health of the environment on demand.
- ▶ Middleware-level updates must be applied in a rolling manner so that maintenance does not require an application outage.

### 6.1.3 The Open Insurance application

Open Insurance is name of the pilot application that Insurance Company X is creating to demonstrate the feasibility of a mobile cloud solution. The Open Insurance application provides a simple interface for customers of Insurance Company X to manage their accounts and file claims for automobile insurance. The application is designed to run on Android and iOS devices.

Open Insurance is being developed in IBM Worklight Studio by using HTML, CSS, and JavaScript. The application does not include any native code, that is, platform-specific code. Instead, it uses a hybrid application structure to achieve native-style performance. Worklight generates the Android and iOS application packages that are based on the hybrid APIs that are used in the Open Insurance application.

The hybrid APIs enable the following useful application features:

- ▶ **Contacts:** Provides information about drivers in the account and makes it easy to automatically contact the customer's agent or Insurance Company X with support questions.
- ▶ **Camera:** Can be used to take pictures to document an accident or to process check payments.
- ▶ **Geolocation:** When an accident is reported, the application can automatically detect the customer's location and dispatch an approved towing service there.



- **Media:** When submitting a claim, the customer can attach recorded witness testimony in audio files.
- **Network:** Provides general connectivity to the company's back-end server infrastructure for authentication and other services, and communicates across the customer's phone service provider's network.
- **Storage:** Hooks into native storage on the customer's device to support account management and claims processing.

When customers load the Open Insurance application, the first thing they see is a login page. The home page of the application provides easy-to-find buttons for important capabilities, such as managing accounts, contacting Insurance Company X, requesting roadside assistance, and submitting accident claims. Figure 5-1 shows these pages of the application.



Figure 6-1 Login and home pages of the Open Insurance application

From the home page of the application, one of the customer's primary options is "Manage your account." In this part of the application, customers can choose to add cars or drivers to their account, check the coverage on their current policy, review claims, access ID cards, and work with various billing options. Under "Billing," customers can view their payment history or make payments by using several common payment options, including personal check (they photograph the check and submit it for electronic processing), credit or debit card, or other forms of online payment. Figure 6-2 shows the application's Billing and Payment Options panels.



Figure 6-2 Billing and Payment Options panels.

The largest workflow in the application is for submitting a new insurance claim. By touching the "I've been in an accident" area on the home page, customers begin the process of creating a car insurance claim. The claim processing logic uses many built-in device functions, such as geolocation, audio recording, maps, and the camera, and integrates that new input with existing information about the customer's account to simplify claims processing.

When customers begin submitting a claim, the Open Insurance application prompts them to allow the application to access their current location. By using such location features, the application can automatically determine each customer's location and the current weather conditions there and include the data in the claim form, as shown in Figure 6-3.



Figure 6-3 Location prompt and sample location-based claims data

When a claim form is updated, customers can specify whether their accident resulted in any injuries and whether more than one vehicle was involved. Like other parts of the application, the form is also linked to each customer's existing account information, which makes it easy to select the car on the policy that was involved in the accident and the authorized driver that was driving, as shown in Figure 6-4. Customers also can use the device's camera to capture photos of any damage, the other driver's license, and so on, and attach the photos to the claim form.



Figure 6-4 Driver-selection panel and accident report form with a car and driver already selected

If towing assistance is needed, the application automatically searches for approved towing providers in the area. Customers can see the approximate wait time for each towing service and then pick the one they want to use. When the claim form is submitted, the towing company dispatches a truck to the location. The form can also be used to locate a nearby body shop to which to transport the vehicle by choosing from a generated list of approved shops in the area or by opening a map to find more options, as shown in Figure 6-5.



Figure 6-5 List of approved towing companies nearby and optional map view for alternative selection

Customers can also use the claim form to record the contact information for any witnesses to the accident and any statements the witnesses want to make. The witness area of the form includes name and telephone contact information, an audio testimony field that uses the device's audio recorder to capture the witness's spoken report, and a signature field with which the witness can use the phone's touchscreen to digitally sign and certify their statement, as shown in Figure 6-6.



Figure 6-6 Witness audio testimony recorder and completed witness testimony form

After completing all of the required fields, customers can submit the Report Accident form to Insurance Company X for processing. Each claim remains available in the Claims area of the application for the customer to review and modify, as needed.

The Open Insurance application uses standard mobile application technologies (HTML, CSS, and JavaScript) with the hybrid APIs in IBM Worklight (such as those to control the device's camera, maps, audio recorder, and network connection) to provide a cross-platform mobile application for customer account and claim management. Insurance Company X uses the Open Insurance application in its cloud-enabled mobile infrastructure pilot project.



## 6.2 Using patterns to cloud-enable a mobile application in the public cloud

To reduce up-front costs and allow it to quickly prove the feasibility of its mobile solution, Insurance Company X decides to start with the IBM Mobile Application Platform Pattern by using IBM SmartCloud Application Workload Services, a public cloud offering from IBM.

This section describes the following tasks that are involved in the public cloud portion of the Open Insurance pilot program:

1. Creating and accessing a SmartCloud Application Workload Service instance.
2. Loading Mobile Application Platform Pattern content into the SmartCloud instance.
3. Defining an environment profile in the SmartCloud instance and configuring IBM Worklight Studio to connect to that instance.
4. Using IBM Worklight Studio to connect to the SmartCloud instance and generate a pattern for the Open Insurance application.
5. Deploying the Open Insurance pattern on the SmartCloud instance, creating an application instance.
6. Updating a deployed application instance in the cloud.
7. Exporting the pilot pattern definition from SmartCloud Application Workload Services to prepare it for eventual use in the private cloud.

### 6.2.1 Creating a SmartCloud Application Workload Services instance

To start its mobile application infrastructure pilot, Insurance Company X purchases public cloud access in IBM SmartCloud Enterprise. The company can try the Mobile Application Platform Pattern that uses a pay-as-you-go model that initially is cheaper than investing in an internal private cloud-based solution.

To prepare the public cloud infrastructure, the mobile application developer at Insurance Company X completed the following steps:

1. After the company purchases access to SmartCloud Enterprise and enrolls its employees as users, the developer logs in to the SmartCloud Enterprise portal at this location:

<https://www.ibm.com/cloud/enterprise>

2. From the portal, the developer goes to the Control Panel and chooses the Service Instances area, where they create a SmartCloud Application Workload Services environment.
3. The developer clicks **Add Service Instance**, chooses a deployment type and location, and then filters the available service offerings that are based on the selected location.

For its project, Insurance Company X chooses an implementation of IBM SmartCloud Application Workload Services 1.1 to ensure compatibility with IBM Mobile Application Platform Pattern 6.0.

Because the company is trying out only simple development use cases, the developer chooses to deploy an instance of IBM SCAWS v1.1 SMALL.



4. After providing basic information about the service instance, such as an instance name, administrator password, and network information, and after reviewing the cost information for the deployment, the developer starts the provisioning process for the service instance by clicking **OK**. This part of the process is performed only when an instance of the cloud provider must be created. If wanted, the developer can use the same cloud provider to deploy multiple copies of the mobile pattern.
5. When the process to provision the service instance is complete, the SmartCloud Enterprise Control Panel displays the IP address information that is needed to connect to the management interface for the SmartCloud Application Workload Services instance, where the developer must perform other steps. By using this information, the developer browses to `http://<IP_ADDRESS_OF_SCAWS>` and then logs in to the management interface by using the default administrator ID (cbadmin) and the password that they specified while they were provisioning the instance. Figure 6-7 shows the SmartCloud Application Workload Services login page.

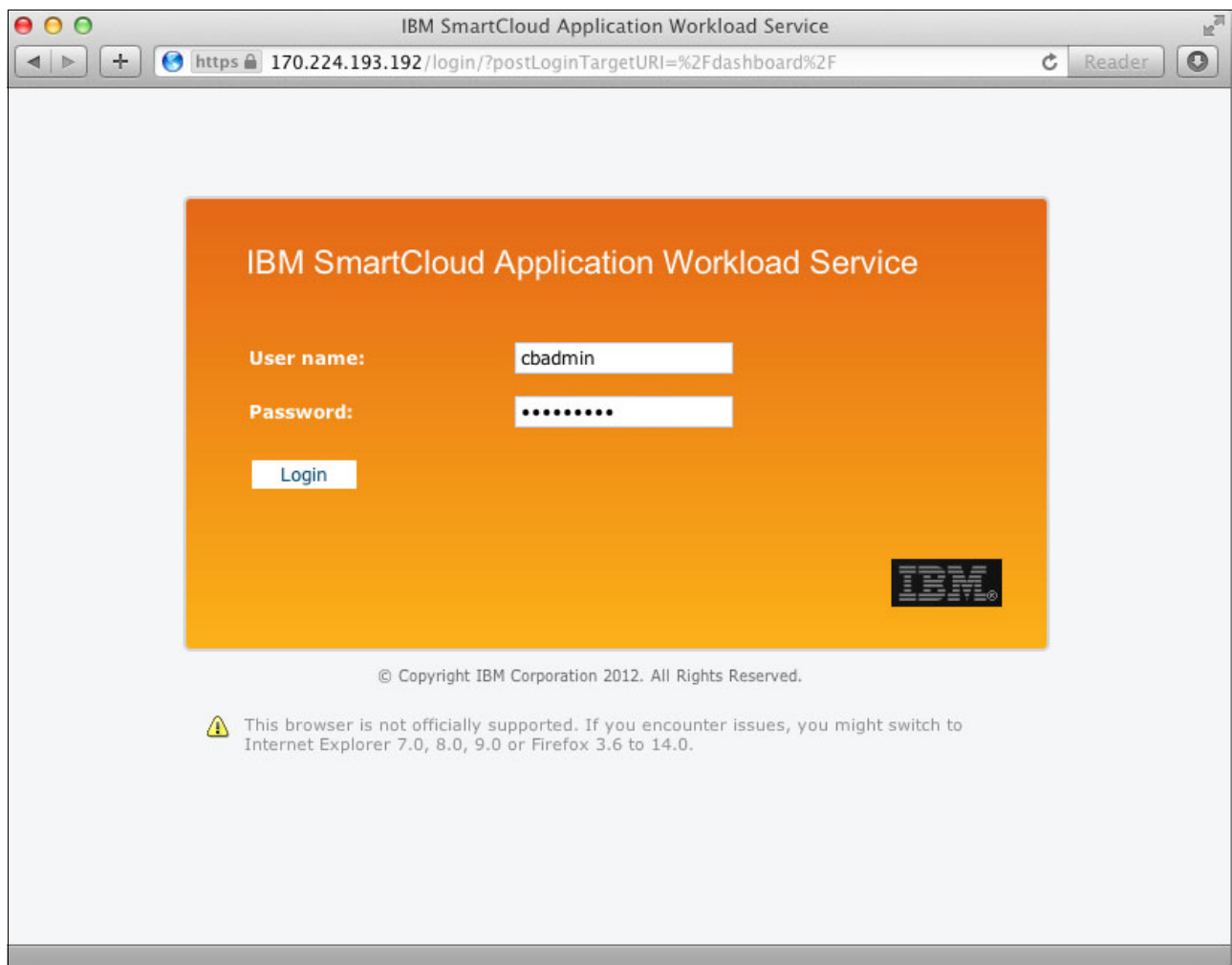


Figure 6-7 Login page of SmartCloud Workload Application Services

## 6.2.2 Loading the Mobile Application Platform Pattern into the SmartCloud instance

By default, SmartCloud Application Workload Services V1.1 contains several patterns, including ones for IBM WebSphere Application Server, IBM DB2, and other products. It does not contain IBM Mobile Application Platform Pattern 6.0, so the developer must load it manually. The pattern features the following components:

- ▶ IBM Mobile Application Platform Pattern 6.0 pattern type
- ▶ Database workload standard for the Worklight reports database
- ▶ Database workload standard for the Worklight runtime database

The developer must now load these three components into the SmartCloud Application Workload Services instance. Because the pattern type is less than 2 GB in size, the developer can load it directly by using the web-based interface that can be accessed by selecting **Cloud** → **Pattern Types**. When the developer loads the new pattern type, they install it under the Local tab and then browse to the location of the pattern type's .TGZ file in their local file system. The developer clicks **OK** to upload it to the cloud provider.

After loading the pattern type, the developer must accept its license and enable the pattern type so that it is available for use.

Figure 6-8 on page 189 shows the listing for the pattern type after the IBM Mobile Application Platform Pattern is loaded but before the developer accepts the license and enables it. After the pattern type is enabled, a green check mark icon is displayed next to its listing, as with the other active pattern types that are shown in Figure 6-8 on page 189.

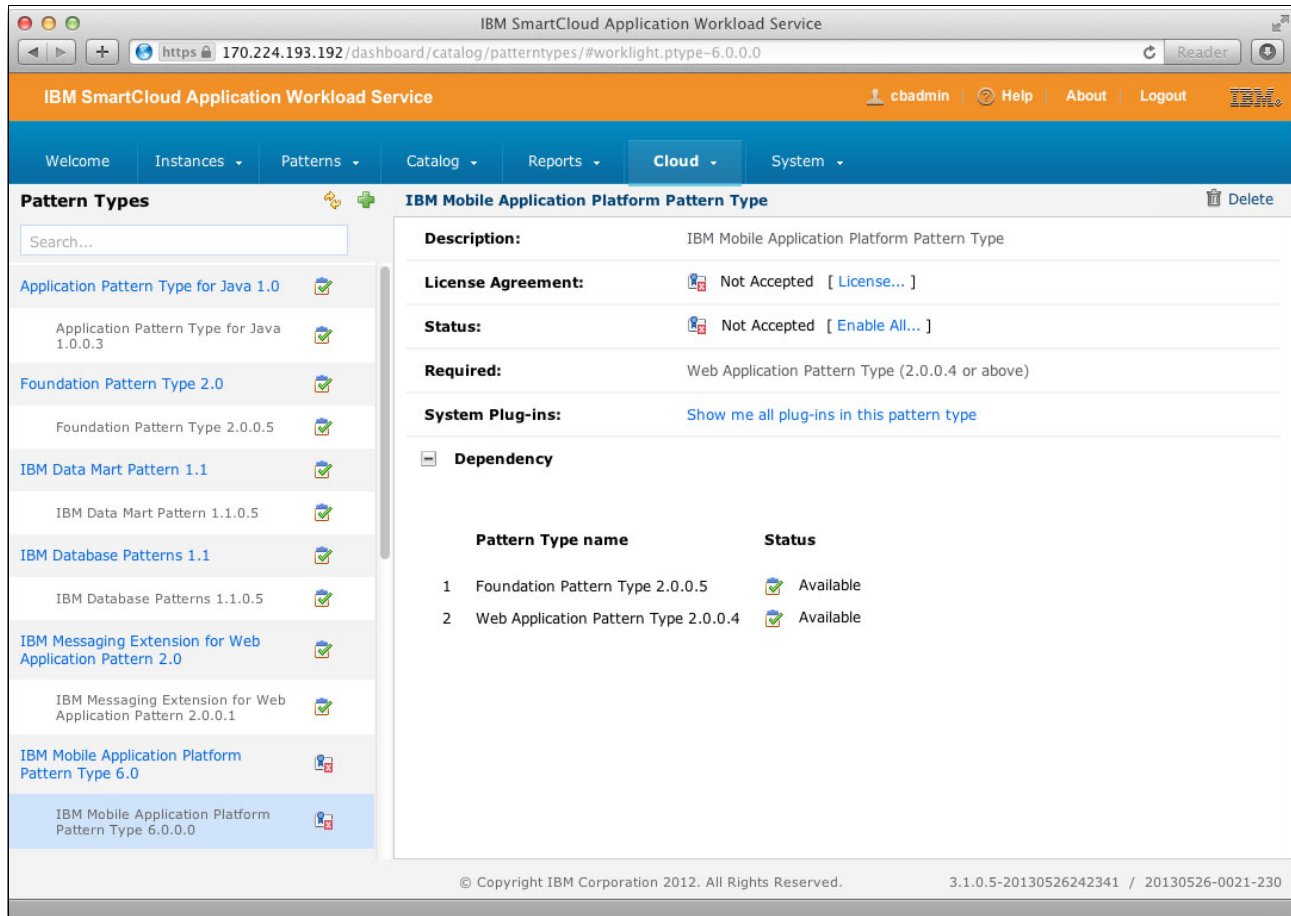


Figure 6-8 Listing for Mobile Application Platform Pattern Type 6.0 in SmartCloud Application Workload Services

Finally, the developer must load both database workload standards by selecting **Catalog** → **Database Workload Standards**. They create two new workload standards, one each for the Worklight runtime and reports databases. The compressed archive files for these database workload standards are packaged with IBM Mobile Application Platform Pattern 6.0.

### 6.2.3 Defining environment profile and connecting Worklight Studio to the SmartCloud instance

The developer is now ready to connect the Worklight Studio mobile application development environment to the cloud provider to create a pattern to enable the Open Insurance application to run on Worklight in the cloud. At the time of this writing, Worklight Studio's cloud integration services were optimized for a private cloud development environment, such as IBM PureApplication System, and do not provide full functionality in SmartCloud Application Workload Services. Therefore, the usage scenario that is described here is for creating the pattern on the SmartCloud Application Workload Services instance only, not for deploying the pattern.

This section describes how to establish the connection from Worklight Studio to the SmartCloud Application Workload Services instance that the developer previously deployed.

To create the Worklight pattern automatically, Worklight Studio must connect to an environment profile on the SmartCloud Application Workload Services instance. An environment profile is a common cloud deployment target that works across many IBM cloud providers and offers a way to create tiered architectures and control resource consumption.

The developer creates the profile by clicking **Cloud** → **Environment Profiles** in SmartCloud Application Workload Services. In the profile configuration, the developer chooses **Pattern Deployer** from the drop-down list that is next to “IP addresses provided by,” so that IP address assignment on deployed instances is handled correctly in SmartCloud Application Workload Services. Then, the developer enables the default cloud group that was created in the SmartCloud instance.

Figure 6-9 shows a sample environment profile.

Mobile profile

Clone Delete

Description:

None provided

Hypervisor type:

SCE

Environment:

All

Created on:

Aug 5, 2013 12:16:39 PM

Current status:

✓

Environment profile can now be used for deployments

Updated on:

Aug 5, 2013 12:17:24 PM

Virtual machine name format:

None provided

IP addresses provided by:

Pattern Deployer

Deploy to cloud groups:

Name	Alias	
SCE Default Cloud	SCE Default Cloud	[remove]

In use	Name	Alias	Subnet address	Gateway	Netmask
<input checked="" type="checkbox"/>	SCE Default IPGrp	SCE Default IPGrp	170.0.0.0	170.224.192.1	255.0.0.0

Add more...

Figure 6-9 Sample environment profile for building a mobile application pattern

In the Worklight Studio workbench, the developer must make sure that the project for the Open Insurance application is loaded into the workspace so that the developer can perform the next step, which is to connect to the public cloud infrastructure to create a pattern for the application.

The developer uses the Preferences menu in Worklight Studio (on OS X, the developer clicks **Eclipse** → **Preferences**; on Windows, the developer clicks **Window** → **Preferences**) to provide connection parameters to the cloud provider. In the Preferences panel, the developer selects **IBM Worklight for IPAS®**. For IPAS Host, the developer uses the IP address of the SmartCloud Application Workload Services instance, provides the required user name and password, and then clicks **Fetch Environment Profiles**. After the profiles are fetched from the remote cloud provider, the developer selects the name of the Profile that was created and then clicks **OK**. Figure 6-10 on page 191 shows the completed window.

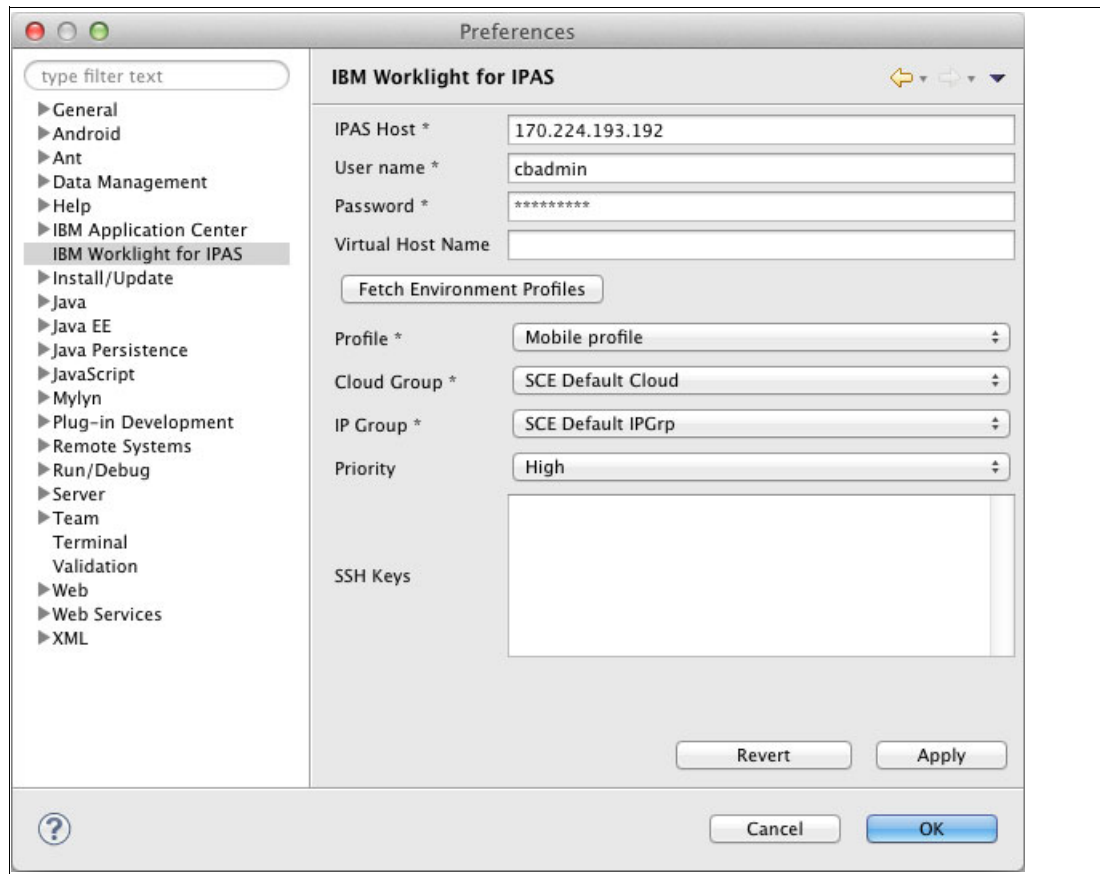


Figure 6-10 Worklight Studio window for connecting the workbench to the cloud provider

## 6.2.4 Generating an Open Insurance pattern from Worklight Studio

Worklight Studio is now connected to the SmartCloud cloud provider and the developer is ready to create the Worklight pattern for the Open Insurance application.

In the main workspace, the developer right-clicks the listing for the application project and then selects **Run As** → **Deploy project to IPAS**. This action connects the workbench to the cloud provider, uploads the application artifacts, and creates the virtual application pattern that is needed to support the Open Insurance application. The pattern is not deployed. A graphical view of the automatically created pattern is shown in Figure 6-11 on page 192.

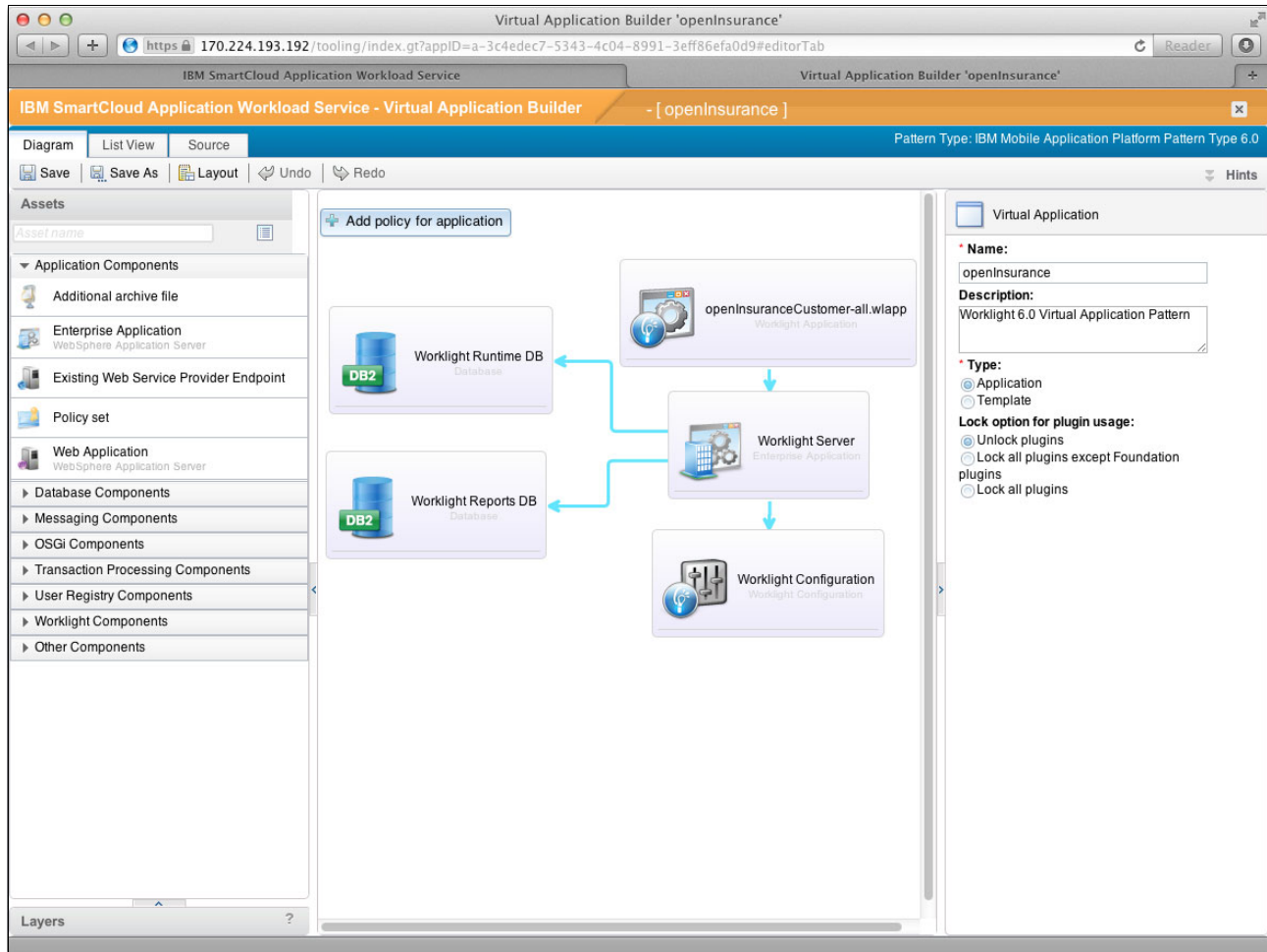


Figure 6-11 Automatically created mobile application pattern

The mobile application pattern includes the following components:

- Worklight server

This component contains WebSphere Application Server and an EAR file that contains Worklight. In this example, Worklight Studio automatically built the required Worklight EAR for the pattern.

- Worklight configuration

This component provides the capability to customize the server configuration and enable administrative security.

- Worklight application

This is the company's new Open Insurance application or, in this case, a Worklight application (WLAPP) package that was automatically built in Worklight Studio and contains the application's server-side components. The application is automatically installed on the Worklight server when the pattern is deployed.

- ▶ Worklight runtime database

This is the database that supports the Worklight runtime components. The scheme and configuration for the database come from the database workload standard that was loaded into the SmartCloud environment when the developer was preparing the pattern components. The database is automatically connected to the Worklight server, with an appropriate database resource reference defined on the cross-component link.

- ▶ Worklight reports database

Similar to the runtime database, this component is automatically configured and connected to the Worklight server. The reports database stores historical report data for the Worklight environment.

## 6.2.5 Deploying a pattern in the public cloud

With the pattern now defined on the SmartCloud Application Workload Service, the developer is ready to deploy it. The developer clicks **Deploy** in the pattern toolbar to open the Deploy Virtual Application window. By using this window, the developer can customize the name and target cloud resources for the deployment and review the estimated cost of the deployment, as shown in Figure 6-12. When ready, the developer clicks **OK** to start the deployment.

Deploy Virtual Application

Name:

IP Version: ☒ IPv4 ☐ IPv6

Cloud group:

☐ Estimate cost of deployment:

Pattern	# Units	Tier	License	Billing	Unit Price
Transactional DB (Worklight_Runtime_DB-db2)	1	Medium	PAYG	Hourly	0.64 USD /UHR
Transactional DB (Worklight_Reports_DB-rh2)	1	Medium	PAYG	Hourly	0.64 USD /UHR

☐ Advanced

Figure 6-12 Pattern deployment dialog in SmartCloud Application Workload Services

During deployment of the pattern, the current instance status is shown on the Virtual Application Instances page. To see this status indicator, the developer clicks **Instances** → **Virtual Applications**, filters on the appropriate pattern type, and selects the instance from the resulting list. The status bar shows how far the deployment progressed, and the History area at the bottom of the window gives a more detailed view of the deployment progression, including time stamps. Deployment time varies slightly based on resource availability and the current workload volume in the public cloud. For the Open Insurance application, it took approximately 48 minutes to deploy and configure the pattern (see Figure 6-13 on page 194), which was well within the 60-minute limit that was established for the pilot.



The screenshot displays the IBM SmartCloud Application Workload Service interface. The top navigation bar includes links for Welcome, Instances, Patterns, Catalog, Reports, Cloud, and System. The main content area is titled 'Virtual Application Instances' and shows a list of instances. The 'openInsurance' instance is selected, and its deployment history is displayed in a table.

History	Timestamp
The virtual system has been deployed	Aug 5, 2013 1:28:24 PM
Script package Must Gather Logs on virtual machine Worklight_Server-was.1137573164 completed successfully	Aug 5, 2013 1:28:24 PM
Executing script package Must Gather Logs on virtual machine Worklight_Server-was.1137573164	Aug 5, 2013 1:28:09 PM
Script package Must Gather Logs on virtual machine Worklight_Reports_DB-db2.113757 completed successfully	Aug 5, 2013 1:28:09 PM
Executing script package Must Gather Logs on virtual machine Worklight_Reports_DB-db2.113757	Aug 5, 2013 1:27:55 PM
Script package Must Gather Logs on virtual machine Worklight_Runtime_DB-db2.113757 completed successfully	Aug 5, 2013 1:27:55 PM
Executing script package Must Gather Logs on virtual machine Worklight_Runtime_DB-db2.113757	Aug 5, 2013 1:27:38 PM
Script package Default add disk on virtual machine Worklight_Reports_DB-db2.113757 completed successfully	Aug 5, 2013 1:27:36 PM
Executing script package Default add disk on virtual machine Worklight_Reports_DB-db2.113757	Aug 5, 2013 1:27:27 PM
Script package Default add disk on virtual machine Worklight_Runtime_DB-db2.113757 completed successfully	Aug 5, 2013 1:27:25 PM
Executing script package Default add disk on virtual machine Worklight_Runtime_DB-db2.113757	Aug 5, 2013 1:27:13 PM
Executing script packages	Aug 5, 2013 1:27:10 PM
Starting virtual machine Worklight_Server-was.1137573164	Aug 5, 2013 12:59:43 PM
Starting virtual machine Worklight_Reports_DB-db2.113757	Aug 5, 2013 12:59:43 PM
Starting virtual machine Worklight_Runtime_DB-db2.113757	Aug 5, 2013 12:59:43 PM
Starting virtual machines in virtual system d-6659142c-f797-4493-8a17-f4a46a392fb3.	Aug 5, 2013 12:59:43 PM
Registering virtual system d-6659142c-f797-4493-8a17-f4a46a392fb3	Aug 5, 2013 12:50:27 PM
Transferring virtual images to hypervisors	Aug 5, 2013 12:50:22 PM
Generating model for topology and network	Aug 5, 2013 12:41:40 PM
Reserving cloud resources	Aug 5, 2013 12:41:06 PM
Deployment has been queued	Aug 5, 2013 12:40:51 PM

© Copyright IBM Corporation 2012. All Rights Reserved. 3.1.0.5-20130526242341 / 20130526-0021-230

Figure 6-13 Virtual Application Instances page that shows the status (Deployment history) of pattern deployment

The deployed virtual application instance contains three virtual machines that were automatically provisioned and configured based on the parameters in the Open Insurance virtual application pattern. Both database virtual machines are running IBM DB2, with the required database tables pre-populated to match the schemas that are needed for the Worklight runtime and reports databases. The Worklight virtual machine is running WebSphere Application Server, with Worklight installed on top of the application server and running the Open Insurance application. This server is also configured with database resource references to the runtime and reports databases in the instance. Figure 6-14 on page 195 shows an example of the instance page with the three virtual machines deployed.

openInsurance

Manage
 Upgrade
 Delete

Name:

openInsurance

Created by:

cbadmin

Started on:

Aug 5, 2013 12:40:45 PM

Access granted to:

ID:

d-6659142c-f797-4493-8a17-f4a46a392fb3

Status:

Running

Using Environment profile:

In cloud group:

SCE Default Cloud

Referenced shared services:

Pattern type:

IBM Mobile Application Platform Pattern Type 6.0

Middleware perspective (5 in total)

WORKLIGHT\_INSTALL (Worklight\_Server-was)

DB2 (Worklight\_Runtime\_DB-db2) [Endpoint](#)

WORKLIGHT (Worklight\_Server-was)

WAS (Worklight\_Server-was) [Endpoint](#)

DB2 (Worklight\_Reports\_DB-db2) [Endpoint](#)

Virtual machine perspective (3 in total)

Name	Public IP	VM Status	Started on	Middleware Status
Worklight_Reports_DB-db2. 11375731645325	170.224.195.58 vhost0823.dc1.on.ca. compute.ihost.com	Running <a href="#">Log</a>	Aug 5, 2013 12:40:53 PM	DB2 <a href="#">Endpoint</a>
Worklight_Runtime_DB-db2. 11375731645324	170.224.194.140 vhost0650.dc1.on.ca. compute.ihost.com	Running <a href="#">Log</a>	Aug 5, 2013 12:40:52 PM	DB2 <a href="#">Endpoint</a>
Worklight_Server-was. 11375731645326	170.224.192.239 vhost0239.dc1.on.ca. compute.ihost.com	Running <a href="#">Log</a>	Aug 5, 2013 12:40:53 PM	WORKLIGHT_INSTALL  WORKLIGHT WAS <a href="#">Endpoint</a>

Figure 6-14 Instance page showing details about the Open Insurance application

The instance page includes a “Virtual machine perspective” section in which the endpoint connections and network information for the deployed virtual machines in the instance are listed. If the developer wants to connect a phone emulator to the Worklight server for testing, for example, they can use the IP address of the Worklight virtual machine to initiate the connection. The Endpoint link for the Worklight server virtual machine opens the IBM Worklight Console, where the developer can see the applications that are running on the server. In this case, there are two running versions of the Open Insurance application, one for iPhone and one for Android, as shown in Figure 6-15 on page 196.

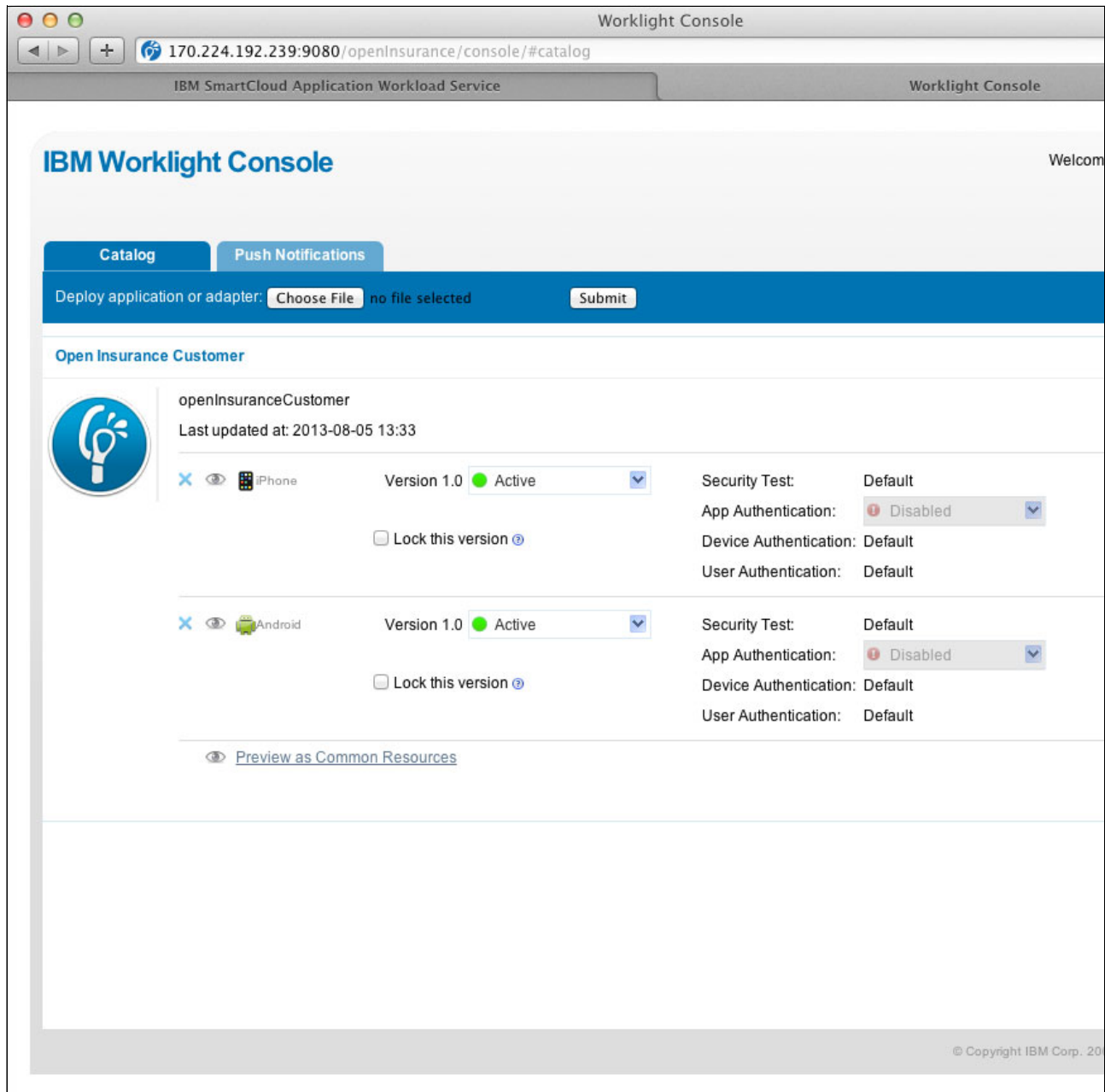


Figure 6-15 Worklight Console showing the installed Open Insurance application versions

Selecting the application link for either platform in Worklight Console opens a new tab with a browser-based application simulator. This is a quick way to verify that the application was installed and is running successfully. Figure 6-16 on page 197 shows the application simulator displaying the login page for the Open Insurance application on an iPhone.

At this point in the pilot, Insurance Company X successfully connected to IBM SmartCloud Application Workload Services, automatically generated a Worklight pattern for its Open Insurance application, deployed the pattern to the public cloud infrastructure, and verified that the application is running.

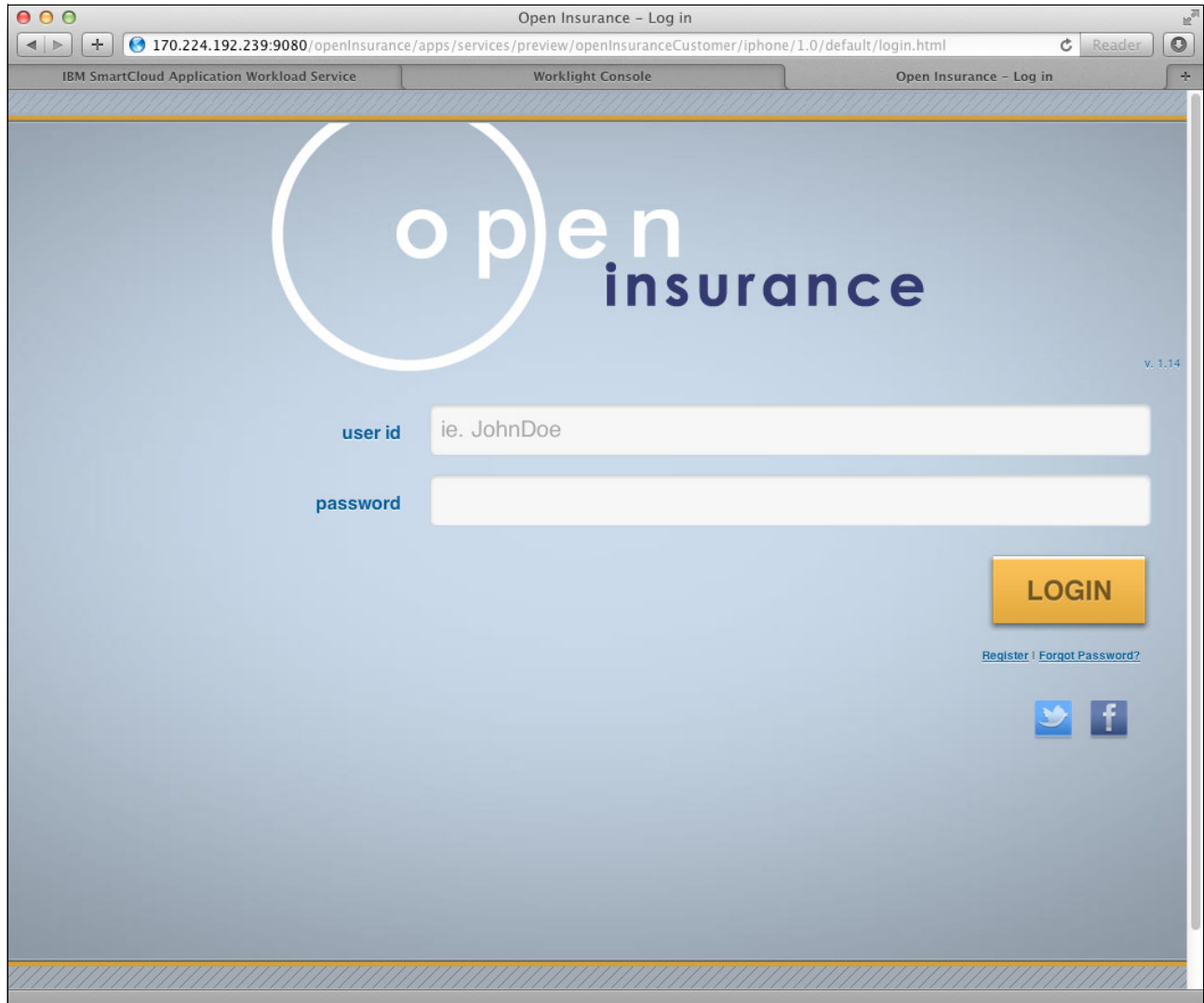


Figure 6-16 Application simulator version of Open Insurance application

## 6.2.6 Rapidly updating a running application instance

With the Open Insurance application now running in the public cloud, the development team at Insurance Company X must evaluate the ease and speed with which the application can be updated.

Every virtual application deployment includes a Virtual Application Console that provides various tools for lifecycle management, including monitoring, logging, and administrative operations. For the Worklight server component, one of the supported operations is to update a deployed application.

To open the Virtual Application Console, the developer goes to the toolbar on the instance page in the SmartCloud console and clicks **Manage**. In the console, the developer goes to the Operations tab and then selects **Worklight server**. From there, the developer has two options: update the application or adapter file for the instance, or work with the security configuration.

In Version 6.0, Worklight Studio is optimized for private cloud use and does not support automatically updating public cloud-deployed applications from the development workbench. Therefore, the developer must start by using Worklight Studio to rebuild the application with the new changes. To rebuild the application, the developer performs a build with a project target of **Deploy to IPAS**, as they did when they were generating the original pattern. Then, the developer pushes the WLAPP application package to the Worklight server by using the Virtual Application Console, which is described next.

To update the application, the developer expands the Worklight Application/Adapter section in the Virtual Application Console. In the Files area, the developer chooses **Edit**. They use the file system browser to find the updated WLAPP application archive to upload to the cloud. In the Insurance Company X scenario, the application change involves only a simple text update for the default user ID string on the application's login page, which is easy to confirm. The original application used JohnDoe as the default user name, and the updated application uses JaneSmith instead.

After the new application package is uploaded to the cloud, the developer clicks **Submit** on the Virtual Application Console. This starts an application update operation, which can be seen at the bottom of the console. Clicking the **Refresh** icon in the Operation Execution Results area updates the operation status information. In the case of the Open Insurance application, after about 5 seconds, the Worklight application update status showed Success (see Figure 6-17 on page 199), which is within the 30-second tolerance window that was identified as a requirement for the pilot.

The screenshot displays the IBM SmartCloud Application Workload Service interface. The top navigation bar includes 'Monitoring', 'Logging', 'Operation' (selected), 'Links', and 'Main Console'. The 'Operation' tab shows a list of components on the left and a detailed view on the right. The detailed view indicates that the 'Worklight Application/Adapter' is launched and shows a 'Security' section with 'Worklight Security'.

Below the operation details, the 'Operation Execution Results' section shows a table with the following data:

Name	Status	Created Time	Result	Return Value
Worklight Application/Adapter	Done	Aug 5, 2013 1:57:18 PM	Worklight_Server-was.11375731645326.WORKLIGHT: Success	Worklight_Server-was.11375731645326.WORKLIGHT: worklight configuration updated successfully

At the bottom of the console, the copyright notice reads: © Copyright IBM Corporation 2012. All Rights Reserved. The version number is 3.1.0.5-20130526242341 / 20130526-0021-230.

Figure 6-17 Results of an application update that is shown in the Virtual Application Console

To verify that the application update was successful, the developer must return to the instance page and load the Endpoint URL for the Worklight virtual machine to return to the Worklight Console. From there, the developer selects one of the application links to load the login page of the application. The default text in the user ID field is now JaneSmith instead of JohnDoe, as expected (see Figure 6-18 on page 200). This means that the application update was successful.

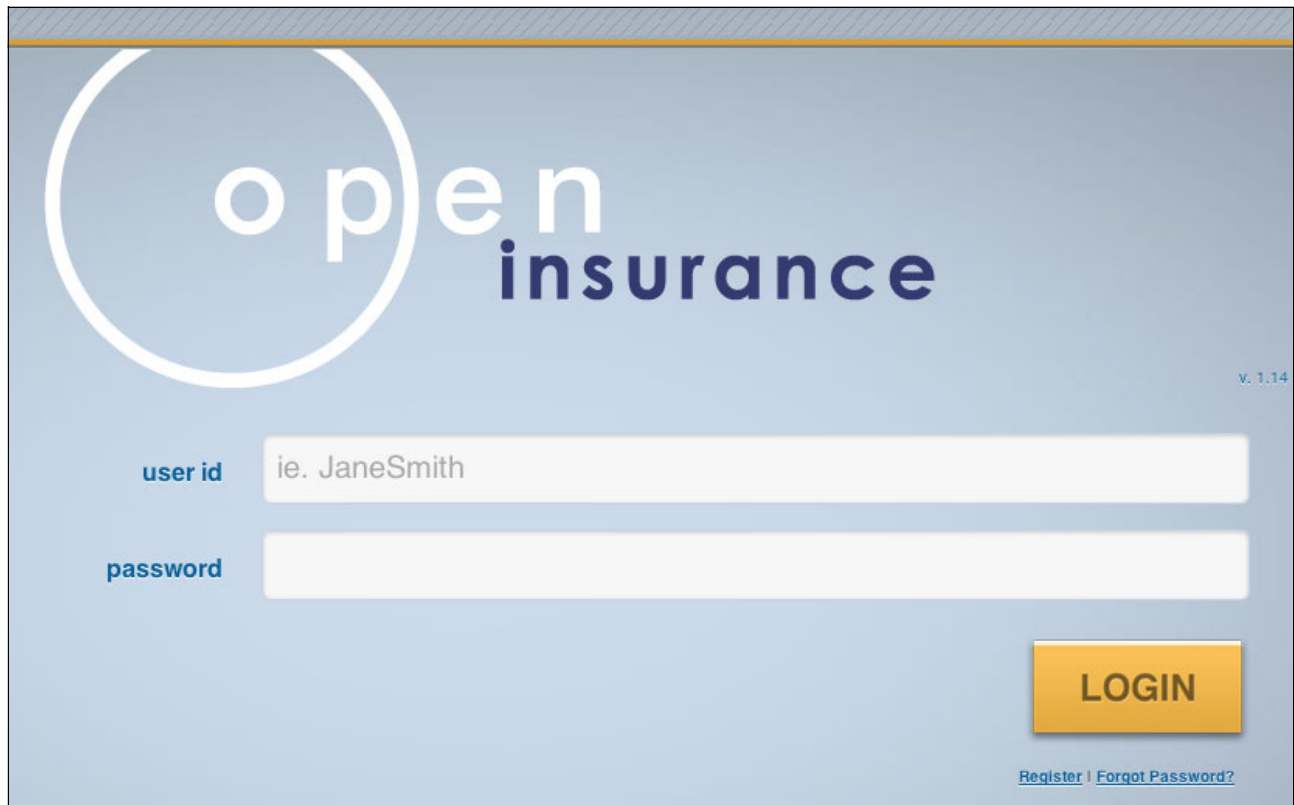


Figure 6-18 Open Insurance application login page that shows updated default user ID text

### 6.2.7 Export pattern content for use in the private cloud

The last criterion for success in the public cloud portion of Insurance Company X's pilot project is to ensure that the application content can be exported from the public cloud for use elsewhere, such as in a private cloud.

Exporting a virtual application pattern from IBM SmartCloud Application Workload Services is simple. The developer starts by using their browser to browse to the virtual application pattern page for the pattern that they want to export; in this case, the Open Insurance pattern (this page is in the SmartCloud console, under **Patterns** → **Virtual Applications**).

The developer then clicks **Export** in the pattern toolbar. The system automatically starts downloading to the developer's workstation a compressed file that contains the pattern definition. The name of the compressed file is the same as the Application ID string for the pattern in the cloud provider. Figure 6-19 on page 201 shows the Open Insurance pattern details page in the SmartCloud console, including the Application ID and pattern toolbar.



openInsurance
Deploy
Open
Export
Delete
Clone

**Application ID:** a-3c4edec7-5343-4c04-8991-3eff86efa0d9

**Description:** Worklight 6.0 Virtual Application Pattern

**Created by:** cbadmin

**Updated by:** cbadmin

**Created on:** Aug 5, 2013 12:26:58 PM

**Updated on:** Aug 7, 2013 11:51:05 AM

**Preview:**

**Access granted to:**

**Pattern type:** IBM Mobile Application Platform Pattern Type 6.0 ( Unlocked )

Figure 6-19 Pattern details page in the SmartCloud console

The downloaded pattern archive file can be imported and used on other cloud providers that support the IBM Mobile Application Platform Pattern 6.0.

## 6.2.8 Public cloud pilot program summary

At this stage, Insurance Company X completed all of the testing scenarios for the public cloud portion of its mobile application pilot project. Table 6-1 shows a summary of the project requirements and results.

Table 6-1 Outcomes of Open Insurance pilot for public cloud

Requirement	Result
A development workbench that integrates with the public cloud	<p>Mostly successful</p> <p>Worklight Studio can connect to the public cloud provider to define patterns, but not deploy them. At the time of this writing, this is a current limitation regarding public cloud integration with Worklight Studio. When connected to a private cloud, Worklight Studio supports end-to-end application updates from the development workbench.</p>

Requirement	Result
Automatic test environment definition on the cloud provider, including all server and database components	<p>Successful</p> <p>Worklight Studio automatically defined the virtual application pattern that is required to deploy the Open Insurance application, including the Worklight server, the Open Insurance application, and the required runtime and reports databases.</p>
Easy deployment of the test environment on the cloud provider	<p>Successful</p> <p>The pilot used SmartCloud Application Workload Services to quickly deploy a pattern and create a test environment. The deployment window took only a few seconds to complete; the use of the window also made it easy to evaluate the estimated cost of each deployment.</p>
Simple tools to develop shared application assets that run on Android and iOS	<p>Successful</p> <p>Worklight provides hybrid APIs that simplify cross-platform application development. The Open Insurance application ran successfully on Android and iOS emulators.</p>
Rapid provisioning of new test servers to the cloud (under 60 minutes)	<p>Successful</p> <p>Provisioning the new virtual application instance for Worklight (three virtual machines) took 48 minutes.</p>
Quick application updates to pre-provisioned servers in the cloud (under 30 seconds) to enable easy testing of code changes	<p>Successful</p> <p>Running an update for the Open Insurance application from the Virtual Application Console took approximately 5 seconds.</p>
An ability to quickly package the application pattern for production deployments	<p>Successful</p> <p>SmartCloud Application Workload Services provided a virtual application pattern export capability that packaged the Open Insurance pattern definition in a compressed archive for use on other cloud providers that support Mobile Application Platform Pattern 6.0.</p>

## 6.3 Transitioning to a private cloud for added flexibility and control

Based on the success of the first phase of its pilot project, where Insurance Company X deployed a development version of its Open Insurance application in a public cloud, the company next wants to explore how to start a production version. Based on a business value analysis, the company believes it can reduce its long-term costs and improve flexibility and control of its application environments by moving to a private cloud implementation.

Unfortunately, the staff at Insurance Company X does not have significant hardware and infrastructure management skills. Therefore, to achieve the expected savings, the company turns to a private cloud solution that offers simple, integrated infrastructure management. The private cloud solution also must support Mobile Application Platform Pattern 6.0 because that is the pattern that is used to deploy the application. Finally, because Insurance Company X plans to deploy all of its application lifecycle environments (development, test, and production) to the new private cloud infrastructure, it needs tight integration between that infrastructure and its mobile application development tools in Worklight Studio.

Ultimately, Insurance Company X chooses to use IBM PureApplication System for its private cloud pilot implementation that is based on considerations, including simple infrastructure management, pattern support, and tool integration.

This section describes the following tasks that are involved in the public cloud portion Open Insurance pilot program:

1. Importing the pattern definition from SmartCloud Application Workload Services into PureApplication System.
2. Making the pattern production-ready by adding policy definitions to enable clustering, failover, scaling, and caching.
3. Deploying the pattern to create a virtual application instance that integrates with the enterprise shared services (Enterprise Load Balancer and Caching) on PureApplication System.
4. Forcing a virtual machine to fail to see automatic failover in action.
5. Driving load against the Worklight servers to force a virtual machine scale-out operation.
6. Applying middleware fixes to the virtual application instance while maintaining access to the running Worklight servers.

These tasks are described next.

### 6.3.1 Importing the Open Insurance pattern into PureApplication System

At the end of the public cloud pilot, the developer at Insurance Company X exported the Open Insurance pattern from the SmartCloud cloud environment. But, before the pattern can be imported into PureApplication System, the developer must load the required pattern content onto the PureApplication System. In this case, that means loading Mobile Application Platform Pattern 6.0 and the two database workload standards for the runtime and reports databases, just as the developer did when they loaded these items into SmartCloud Application Workload Services.

To import the pattern definition into PureApplication System, the developer goes to the Workload Console for PureApplication System, selects **Patterns** → **Virtual Applications**, and then selects **IBM Mobile Application Platform Pattern Type 6.0** from the filter list that is on the left side of the window. The developer then clicks **Import** and browses to find the compressed archive of the Open Insurance application that was exported from SmartCloud Application Workload Services.

When the developer selects the archive from the file system window, the pattern archive is automatically uploaded to the Workload Console and becomes available on PureApplication System. Because the configuration object IDs that are associated with the pattern components are likely to vary from one cloud provider to another, the leading practice is to open the pattern and verify that all of its connections are valid, there are no errors, the database workload standards are mapped correctly, the resource references are correct, and so on. Figure 6-20 shows an example of the pattern as viewed PureApplication System.

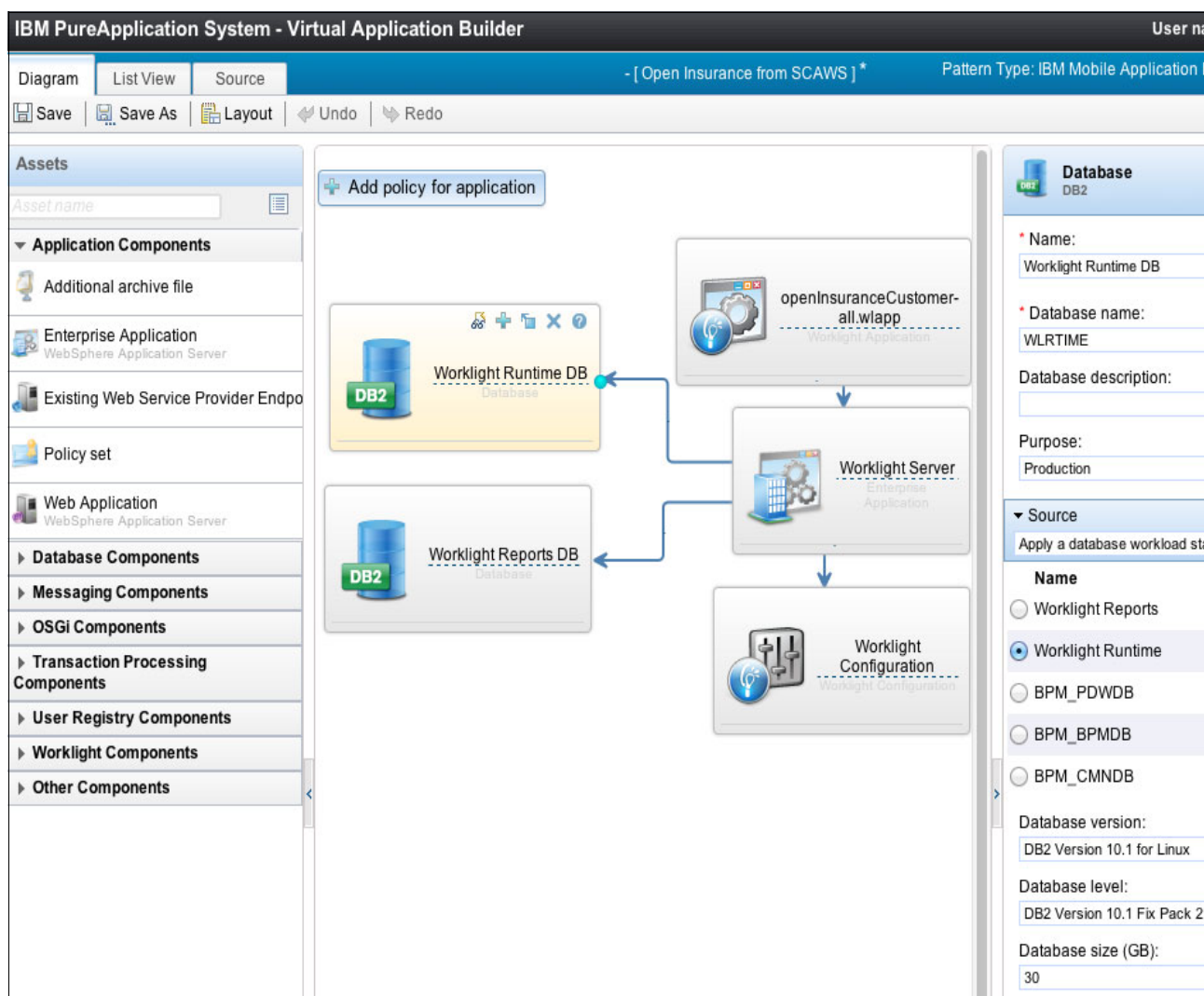


Figure 6-20 Virtual application pattern for Open Insurance in PureApplication System

The developer now can deploy a test copy of the pattern and it behaves identically to the instance in SmartCloud Application Workload Services. However, for its private cloud implementation, Insurance Company X wants to enable production-level qualities of service with provisions for clustering, failover, caching, and scaling. This can be achieved in virtual application patterns by using policy-based configurations.

### 6.3.2 Making the pattern production-ready

To enable clustering, failover, scaling, and caching, the developer must add the required policies to the Open Insurance pattern.

The developer starts by opening the pattern in the pattern editor in the PureApplication System console. Then, they add a Scaling Policy to the pattern by selecting the Worklight server component in the pattern canvas and then clicking the plus sign icon (+) open the Policy menu. The developer then chooses **Scaling Policy**. Based on the parameters that are defined in the solution requirements, the developer configures the following scaling policy options:

- ▶ Selects **Enable session caching**
- ▶ Sets the **2 - 10** instance range
- ▶ Selects the **CPU Based** scaling and sets the range to **20% - 80%**, with a threshold period of **120** seconds
- ▶ Chooses to scale by using the **add or remove nodes** option

To enable routing across a cluster of Worklight servers by using a common virtual host definition, the developer also adds a Routing Policy to the application pattern and specifies a Virtual Host Name. Now, any deployed copy of the pattern includes a minimum of two Worklight server instances that are connected to a caching service for HTTP session caching, with automatic failover and elastic scaling when CPU usage falls outside of the specified ranges. Figure 6-21 on page 206 shows the pattern with the latest policy changes.

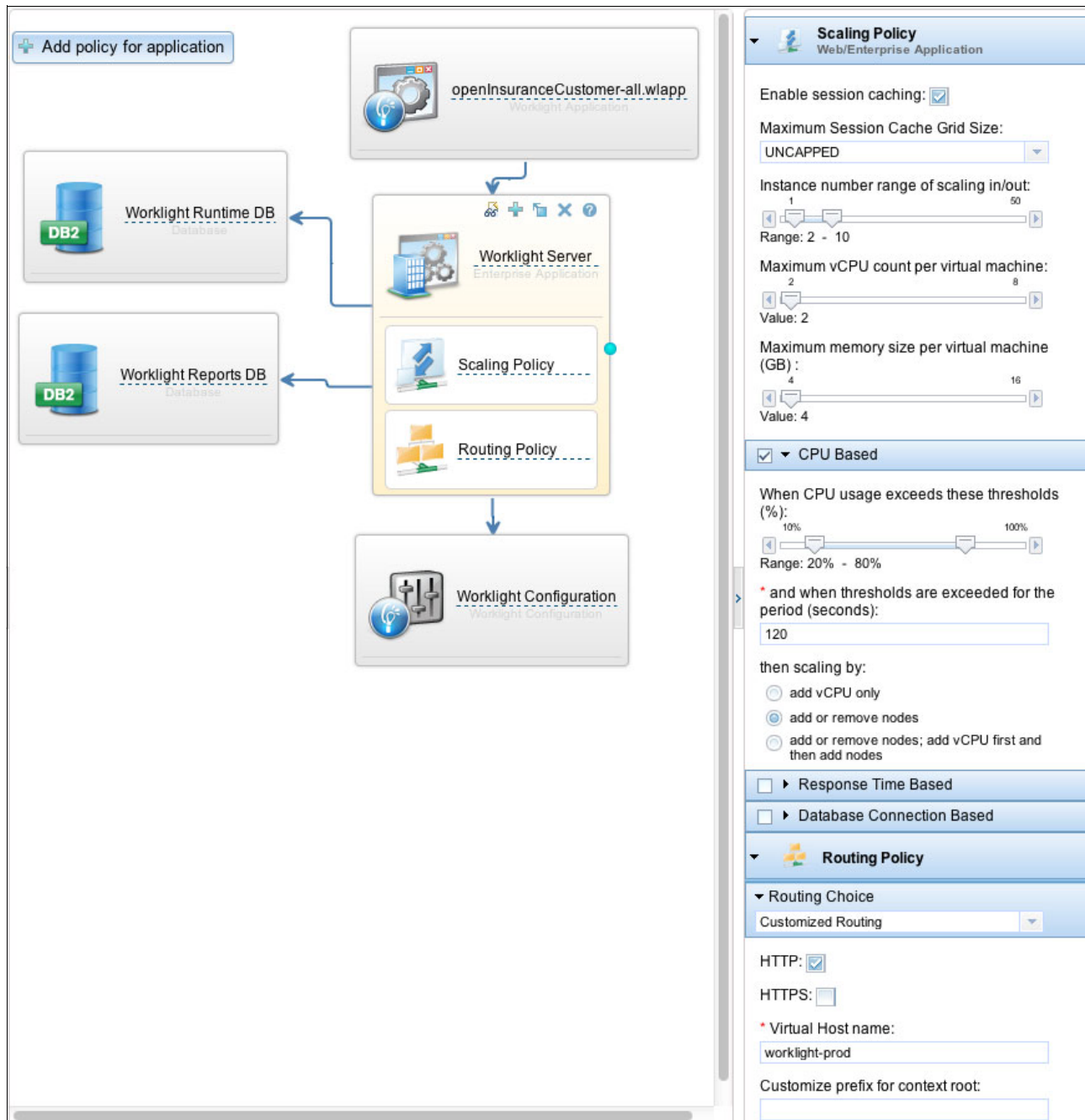


Figure 6-21 Open Insurance virtual application pattern with scaling and routing policies that are enabled

### 6.3.3 Deploying the pattern to automatically integrate with PureApplication System's enterprise shared services

In 6.3.2, "Making the pattern production-ready" on page 205, the developer used policies to enable clustering, failover, scaling, and caching, which quickly transformed the original Open Insurance pattern into an enterprise-enabled application pattern. Now that the pattern definition is enhanced, it is time to deploy the pattern and see the enterprise-level qualities of service in action.

Before the enterprise-enabled Open Insurance pattern can be deployed, the required shared services already must be deployed in the target cloud group for the specified pattern. Whenever a scaling policy or a routing policy is used, the Enterprise Load Balancer (ELB) Proxy Service must be active. Because the scaling policy that was applied specifies a connection to a session cache, the Caching Service also must be active.

To ensure that these requirements are met, the developer at Insurance Company X must confirm that the required services are available in the target cloud group. The developer does this by clicking **Instances** → **Shared Services** and verifying that the Caching Service and the ELB Proxy Service are present on all deployed instances, as shown in Figure 6-22.

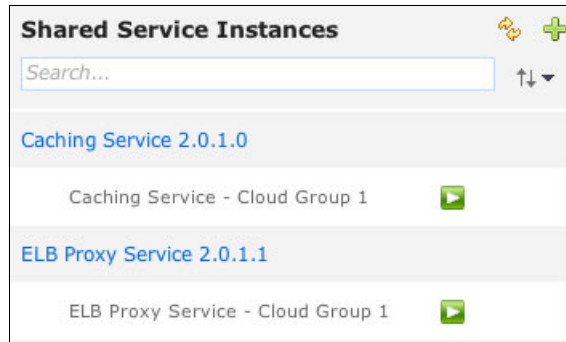


Figure 6-22 Caching Service and ELB Proxy Service confirmation

Deploying patterns in PureApplication System (private cloud) follows the same procedure as deploying patterns in SmartCloud Application Workload Services (public cloud). For this private cloud portion of the pilot, the developer goes to the PureApplication System Workload Console, selects the pattern, and clicks **Deploy** in the pattern toolbar. The developer selects a target Environment Profile and Cloud Group that have sufficient capacity for the deployment and where the required shared services are active. Later, the developer must connect into the virtual machine by using a Secure Shell (SSH) terminal for a failover test. The developer can specify an SSH key now (under Advanced) or provide the key later in the Virtual Application Console.

After the pattern is deployed, the instance page in the Workload Console includes “Referenced shared services” for the caching service and the proxy service. This means that the Worklight server is correctly wired into these services to provide common routing, failover, elasticity, and session caching, as shown in Figure 6-23 on page 208. The virtual machine perspective for this instance includes four virtual machines, two for DB2 and two clustered instances of the Worklight server, as shown in Figure 6-24 on page 208. This instance includes two Worklight servers by default, which are based on the minimum number of servers that are configured in the scaling policy. This provides high availability in the deployed virtual application instance.



Open Insurance for Production
Stop Start Manage Maintain Resume

<b>Name:</b>	Open Insurance for Production
<b>Created by:</b>	admin
<b>Started on:</b>	Aug 7, 2013, 5:49:07 PM
<b>Access granted to:</b>	Default Admin [ all ] <input type="text" value="Add more..."/>
<b>ID:</b>	d-a85b6f87-5a02-45a7-a1db-6d6e15bfb75f
<b>Status:</b>	Running
<b>Using Environment profile:</b>	<a href="#">T3 Profile</a>
<b>Priority:</b>	High
<b>In cloud group:</b>	<a href="#">Cloud Group 1</a>
<b>Referenced shared services:</b>	<a href="#">Caching Service</a> <a href="#">ELB Proxy Service</a>

Figure 6-23 Shared services for caching and proxy services that are referenced in deployed instance

Open Insurance for Production
Stop Start Manage Maintain Resume Delete

Virtual machine perspective (4 in total)

Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports-DB-db2-11375922947921</a>	9.55.60.10 vm010.dub.usoh.ibm.com	Running <a href="#">Log</a>	Aug 7, 2013, 5:49:28 PM	DB2 <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime-DB-db2-11375922947920</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running <a href="#">Log</a>	Aug 7, 2013, 5:49:27 PM	DB2 <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was-11375922947922</a>	9.55.60.11 vm011.dub.usoh.ibm.com	Running <a href="#">Log</a>	Aug 7, 2013, 5:49:28 PM	WORKLIGHT_INSTALL WORKLIGHT WAS <a href="#">Endpoint</a> ElbServicePlaceholder <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was-21375922947923</a>	9.55.60.13 vm013.dub.usoh.ibm.com	Running <a href="#">Log</a>	Aug 7, 2013, 5:49:29 PM	WORKLIGHT_INSTALL WORKLIGHT WAS <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>

Figure 6-24 Deployed instance with Worklight virtual machines deployed as a cluster with a minimum of two servers

To check the application deployment, the developer accesses the Worklight Console by using the Endpoint link for either of the Worklight server virtual machines. In this private cloud deployment, the console's Endpoint link is formatted to point to the default virtual host that was defined in the routing policy in the application pattern. To resolve that host, the developer must update the configuration on their workstation to map the default host to an instance IP address in the proxy shared service. In a typical production environment, this network mapping is handled at the enterprise level by the data center's network engineers rather than at the desktop level.

### 6.3.4 Forcing a virtual machine failure to see automatic failover in action

To test the failover capabilities of the instance, the developer must use SSH into one of the Worklight virtual machines and force a kernel panic by running some simple commands. The developer must use SSH to connect to the virtual machine by using SSH keys, so if the virtual machine is not already configured with a key, the developer must add one in the Virtual Application Console.

To do this, the developer goes to the instance page for the application, locates the IP address of one of the Worklight virtual machines in the instance, and use SSH to connect to the host as the user ID `virtuser` by using the defined key. Then, after they are connected to the host, the developer runs the following commands:

```
sudo su
echo 1 > /proc/sys/kernel/sysrq
echo o > /proc/sysrq-trigger
```

The commands cause a kernel panic in the virtual machine, which then becomes unresponsive. After a few minutes, PureApplication System detects that the virtual machine failed and automatically begins deploying a new Worklight server in the cluster. This action is a result of the scaling policy that the developer established in the pattern to specify that there should be at least two Worklight servers running always. Figure 6-25 on page 210 shows that the original virtual machine was detected as being stopped, and shows a new Worklight virtual machine being started in the instance.

After this failover operation is complete, there are two running Worklight virtual machines in the instance, as shown in Figure 6-26 on page 210. During failover, the application environment maintains session affinity because HTTP session information is automatically stored in the shared caching service. The failed virtual machine is not deleted, so the developer still can review the logs to determine the cause of failure and attempt to restart the server.

Virtual machine perspective (5 in total)					
Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports DB-db2. 11375922947921</a>	9.55.60.10 vm010.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:28 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime DB-db2. 11375922947920</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:27 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 11375922947922</a>	9.55.60.11 vm011.dub.usoh.ibm.com	Stopped	Aug 7, 2013, 5:49:28 PM		<a href="#">View</a>
<a href="#">Worklight Server-was. 21375922947923</a>	9.55.60.13 vm013.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:29 PM	ElbServicePlaceholder WORKLIGHT_INSTALL WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 31375930791983</a>	9.55.60.15 vm015.dub.usoh.ibm.com	Starting	Aug 7, 2013, 7:59:53 PM		<a href="#">View</a>

Figure 6-25 Instance page in the Workload Console that shows the triggered failover operation

Virtual machine perspective (5 in total)					
Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports DB-db2. 11375922947921</a>	9.55.60.10 vm010.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:28 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime DB-db2. 11375922947920</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:27 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 11375922947922</a>	9.55.60.11 vm011.dub.usoh.ibm.com	Stopped	Aug 7, 2013, 5:49:28 PM		<a href="#">View</a>
<a href="#">Worklight Server-was. 21375922947923</a>	9.55.60.13 vm013.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:29 PM	ElbServicePlaceholder WORKLIGHT_INSTALL WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 31375930791983</a>	9.55.60.15 vm015.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 7:59:53 PM	ElbServicePlaceholder WORKLIGHT_INSTALL WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>

Figure 6-26 After a new virtual machine is added to the instance, the failed VM remains for troubleshooting

### 6.3.5 Drive load against the application to see scaling in action

To show the elasticity of the Worklight cluster, the developer must drive CPU usage of the servers above 80% for approximately 2 minutes, as defined in the scaling policy. For the private cloud pilot of Insurance Company X, the developer decided to use JMeter to drive up usage by sending a large volume of HTTP requests to the application.

**Note:** For testing purposes, it is sometimes also useful to set an artificially low scaling threshold to make it easier to drive a load sufficient to exceed the limit.

As the developer drives load against the application, they also monitor the health of the environment by using the Virtual Application Console. The Monitoring tab on the console includes data for virtual machine monitoring (for example, network traffic and processor usage, as shown in Figure 6-27) and middleware monitoring (for example, heap usage, web request information, and connection pool data, as shown in Figure 6-28 on page 212). After the load to the Worklight cluster crosses the defined threshold, PureApplication System automatically scales up the cluster by adding a third Worklight server, as shown in Figure 6-29 on page 213. If traffic to the cluster later drops below the minimum threshold, PureApplication System automatically scales back the cluster to release the unneeded resources back into the shared cloud pool.

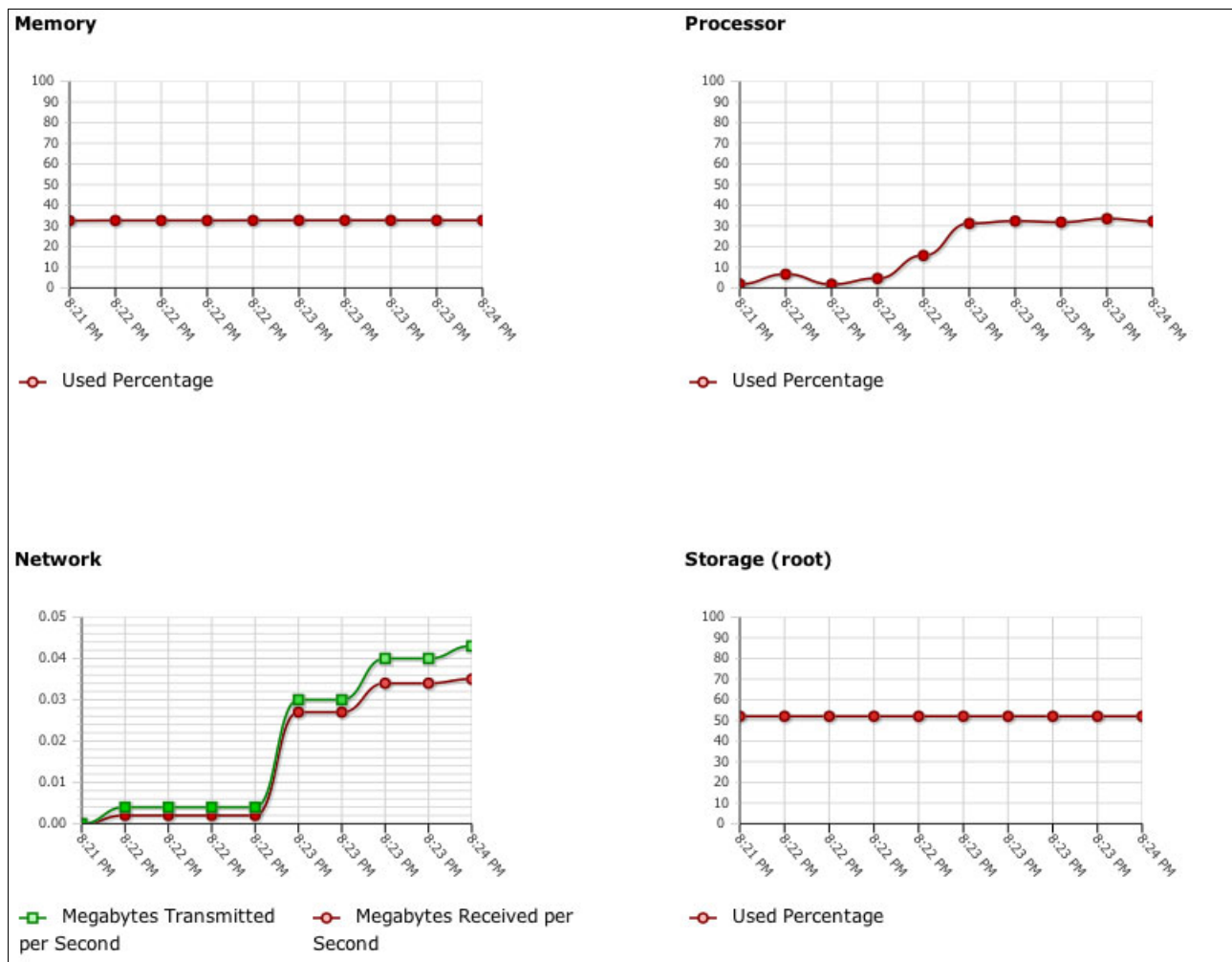


Figure 6-27 Virtual machine monitoring panel in the Virtual Application Console

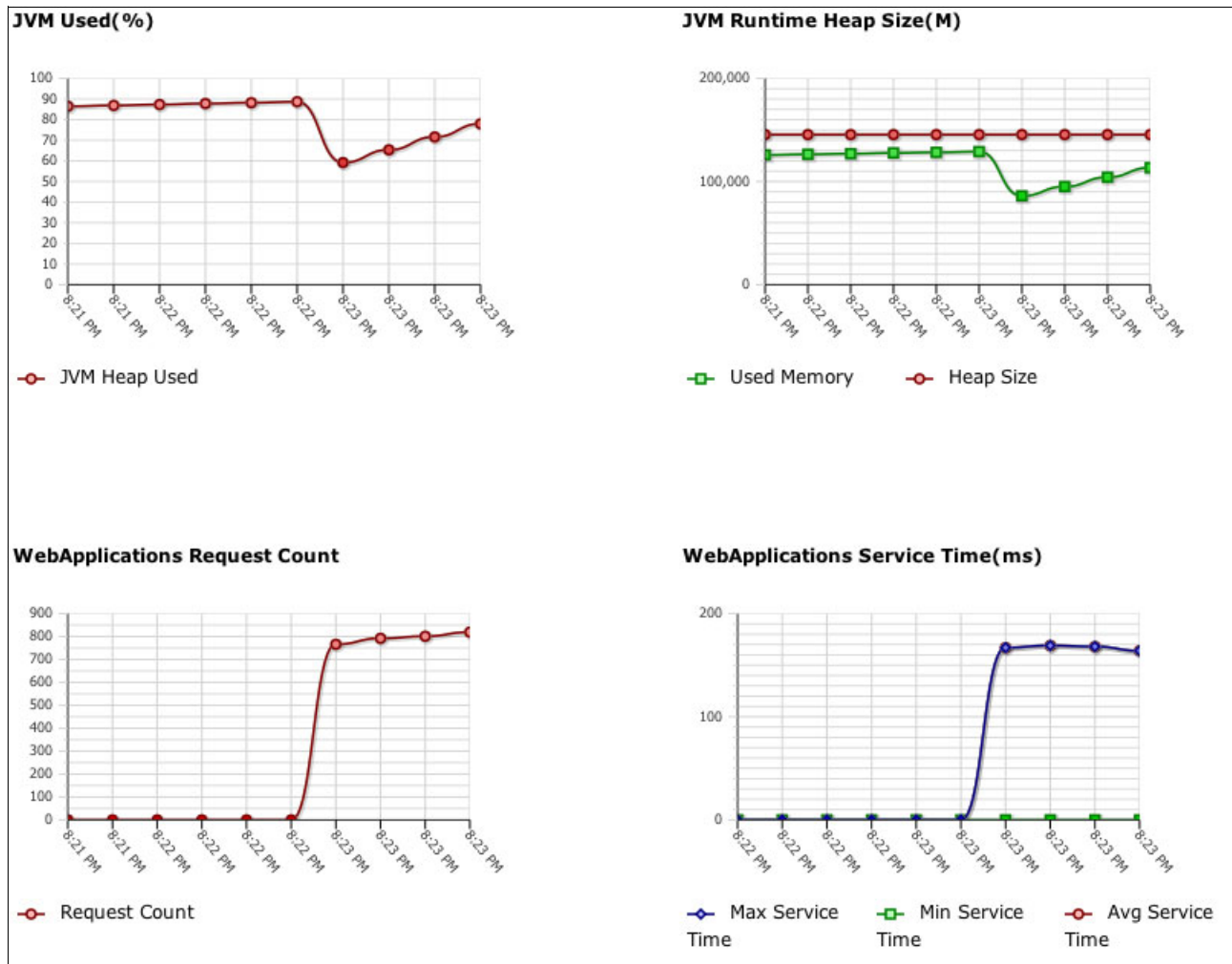


Figure 6-28 Middleware monitoring panel in the Virtual Application Console

Virtual machine perspective (6 in total)					
Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports DB-db2. 11375922947921</a>	9.55.60.10 vm010.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:28 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime DB-db2. 11375922947920</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:27 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 11375922947922</a>	9.55.60.11 vm011.dub.usoh.ibm.com	Stopped	Aug 7, 2013, 5:49:28 PM		<a href="#">View</a>
<a href="#">Worklight Server-was. 21375922947923</a>	9.55.60.13 vm013.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 5:49:29 PM	ElbServicePlaceholder WORKLIGHT_INSTALL WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 31375930791983</a>	9.55.60.15 vm015.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 7:59:53 PM	ElbServicePlaceholder WORKLIGHT_INSTALL WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 41375932406244</a>		Launching			<a href="#">View</a>

Figure 6-29 Instance page in the Workload Console that shows PureApplication System adding a virtual machine

At this stage, Insurance Company X showed that it can deploy its Worklight server in a clustered topology with automatic high availability and elasticity. However, there is more to prove in the company's private cloud pilot project.

### 6.3.6 Applying middleware-level fixes to the virtual application instance

The last use case for Insurance Company X's pilot is applying middleware maintenance. In this scenario, the developers are assuming that a critical, immediate fix for WebSphere Application Server must be applied to the Worklight environment.

The first step in applying middleware maintenance to deployed instances is to load the fix into the PureApplication System catalog. The developer clicks **Catalog** → **Emergency Fixes** and in the "Emergency fix files" area, uploads the fix file from their local workstation. In the pilot scenario, this is a standard iFix archive from IBM Fix Central, so no special processing or formatting is required to use it in PureApplication System.

For this fix to be available to apply to deployed patterns on PureApplication System, the developer must configure the emergency fix to be applicable to the relevant pattern content. To do this, the developer adds in the Applicable to field the plug-ins that this fix can be applied to (in this case, WebSphere Application Server). From the Plug-ins area, the developer selects the WebSphere Application Server component (which is labeled "was" in the interface) of the Web Application Pattern Type.

When the developer attempts to apply this fix from the Virtual Application Console, it is present in the list of available fixes. Figure 6-30 shows an example of an emergency fix in the catalog.

8.0.0.0-ws-was-ifpm80935

Refresh
 Delete

<b>Description:</b>	None provided
<b>Created on:</b>	Jul 10, 2013, 11:03:54 AM
<b>Updated on:</b>	Jul 10, 2013, 11:04:15 AM
<b>Emergency fix files:</b>	<div>Browse...</div> <div>Upload</div> <div>The script package is in 8.0.0.0-ws-was-ifpm80935.zip.  Download</div>
<b>Environment:</b>	(none)
<b>Access granted to:</b>	<div>Everyone [read] [remove]</div> <div>admin [owner]</div> <div>Add more...</div>
<b>Severity:</b>	Normal
<b>Applicable to:</b>	<div> <b>Images:</b> <div>Add more...</div> </div> <div> <b>Plugins:</b> <div>was/2.0.1.0 [remove]</div> <div>Add more...</div> </div>

Figure 6-30 WebSphere Application Server fix in the PureApplication System catalog

Next, the developer uses the Virtual Application Console to apply the fix to the Worklight cluster. From the Operations tab, the developer selects the WebSphere Application Server component and in the **Fundamental** area, the developer expands the Update configuration section. In the Interim fixes URL area, the developer uses the Select menu to choose the appropriate emergency fix (or fixes; multiple fixes can be applied at the same time), and then clicks **Submit** to start the update. The status of the maintenance task is displayed in the Operation Execution Results area at the bottom of the Virtual Application Console, as shown in Figure 6-31 on page 215.



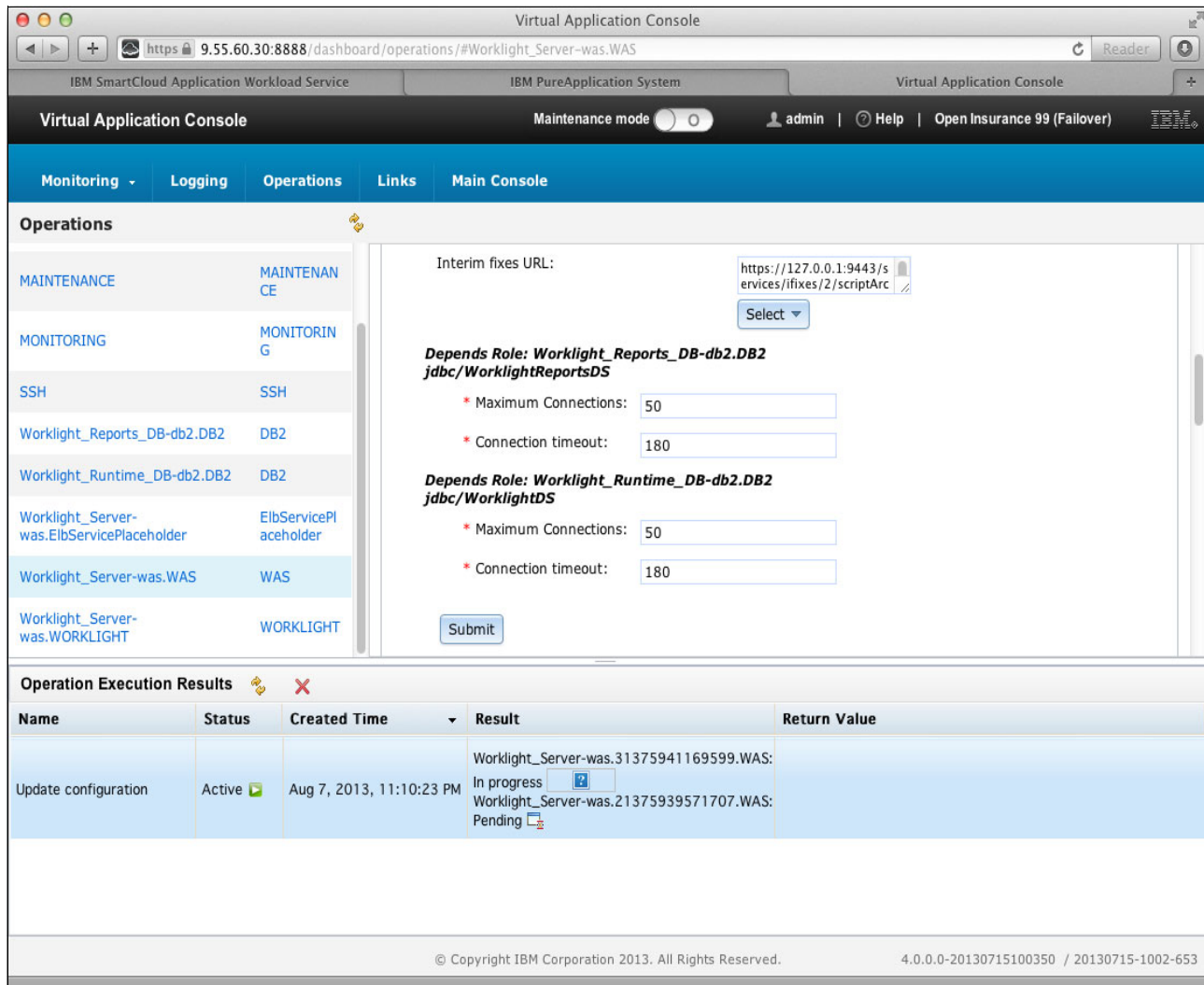


Figure 6-31 Applying an emergency fix to the deployed virtual application instance from the Virtual Application Console

To apply the WebSphere fix, PureApplication System performs a rolling restart of the WebSphere process in each virtual machine and then applies the fix to each machine in sequence. This method for maintaining a clustered environment ensures that at least one of the Worklight virtual machines always remains active so Insurance Company X can maintain server-side application availability, even during a middleware maintenance cycle. Figure 6-32 on page 216 and Figure 6-33 on page 216 show the virtual application instance page during a middleware maintenance operation.

**Note:** The Endpoint link is active and available for one server while the fix is still being applied to the other server.

Virtual machine perspective (5 in total)					
Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports DB-db2. 11375939571705</a>	9.55.60.28 vm028.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:33 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime DB-db2. 11375939571704</a>	9.55.60.27 vm027.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:33 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 11375939571706</a>	9.55.60.29 vm029.dub.usoh.ibm.com	Stopped	Aug 7, 2013, 10:26:34 PM		<a href="#">View</a>
<a href="#">Worklight Server-was. 21375939571707</a>	9.55.60.30 vm030.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:34 PM	WORKLIGHT_INSTALL	<a href="#">View</a>
				WORKLIGHT WAS → <a href="#">Endpoint</a> ElbServicePlaceholder <a href="#">Show more</a>	
<a href="#">Worklight Server-was. 31375941169599</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:52:50 PM	ElbServicePlaceholder WORKLIGHT_INSTALL	<a href="#">View</a>
				WORKLIGHT WAS <a href="#">Show more</a>	

Figure 6-32 Console showing maintenance that is initially applied to the second Worklight virtual machine

Virtual machine perspective (5 in total)					
Name	Public IP	VM Status	Started on	Middleware Status	Action
<a href="#">Worklight Reports DB-db2. 11375939571705</a>	9.55.60.28 vm028.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:33 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Runtime DB-db2. 11375939571704</a>	9.55.60.27 vm027.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:33 PM	DB2 → <a href="#">Endpoint</a> <a href="#">Show more</a>	<a href="#">View</a>
<a href="#">Worklight Server-was. 11375939571706</a>	9.55.60.29 vm029.dub.usoh.ibm.com	Stopped	Aug 7, 2013, 10:26:34 PM		<a href="#">View</a>
<a href="#">Worklight Server-was. 21375939571707</a>	9.55.60.30 vm030.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:26:34 PM	WORKLIGHT_INSTALL WORKLIGHT WAS	<a href="#">View</a>
				ElbServicePlaceholder <a href="#">Show more</a>	
<a href="#">Worklight Server-was. 31375941169599</a>	9.55.60.9 vm009.dub.usoh.ibm.com	Running → <a href="#">Log</a>	Aug 7, 2013, 10:52:50 PM	ElbServicePlaceholder WORKLIGHT_INSTALL	<a href="#">View</a>
				WORKLIGHT WAS → <a href="#">Endpoint</a> <a href="#">Show more</a>	

Figure 6-33 Console showing maintenance actions on first Worklight virtual machine (other fixes now complete)

### 6.3.7 Private cloud pilot project summary

At this stage, Insurance Company X completed all of the planned use cases for its second, private cloud pilot project, which involved the creation of a production environment for its new mobile application. Table 6-2 shows a summary of the project's results.

*Table 6-2 Outcomes of private cloud pilot for Open Insurance application*

Requirement	Result
The application must run in a clustered configuration, with a minimum of two Worklight servers running always.	Successful  The scaling policy in the pattern forced the deployed instance to run the Worklight server in a clustered configuration with at least two servers.
If there is a Worklight server failure, the cloud provider must automatically detect the failure and add a new server to the cluster; the new server should be available within 15 minutes of the initial failure.	Successful  After forcing a virtual machine failure, PureApplication System detected the failed machine and added a new one to the instance to take over. From the time of the initial failure, it took 11 minutes for the new virtual machine to start.
If CPU usage across the cluster exceeds 80% for more than 2 minutes, the cloud provider must add extra server capacity to bring usage below the threshold.	Successful  By defining the “80% for more than 2 minutes” requirement in the scaling policy, PureApplication System automatically provided the elasticity that is needed in the deployed environment.
If CPU usage across the cluster drops below 20% for more than 2 minutes, the cloud provider must remove servers from the cluster and return the infrastructure resources to the shared cloud pool.	Successful  As with CPU usage, by defining the “20% for more than 2 minutes” requirement in the pattern’s scaling policy, elasticity was provided automatically.
HTTP session caching must be provided so that, if there is a server failure, the environment can maintain session affinity for active requests.	Successful  The scaling policy in the application also included a parameter to hook the deployed instance into a shared caching service to provide HTTP session caching. This was enabled with minimal configuration (selecting the option in the pattern).
Basic monitoring for the virtual hardware and middleware must determine the overall health of the environment on demand.	Successful  The Virtual Application Console includes basic monitoring tools for the virtual machine and middleware. These were helpful during the scaling scenario to monitor the health and performance of the environment.
Middleware-level updates must be applied in a rolling manner so that maintenance does not require an application outage.	Successful  The Virtual Application Console includes a simple middleware update option for standard iFixes. These fixes were applied in a rolling fashion.



# Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some of the publications that are referenced in this list might be available in softcopy only:

- ▶ *Adopting IBM PureApplication System V1.0*, SG24-8113
- ▶ *Creating Composite Application Pattern Models for IBM PureApplication System*, SG24-8146
- ▶ *IBM Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud*, SG24-8011
- ▶ *Creating Smart Virtual Appliances with IBM Image Construction and Composition Tool*, SG24-8042:
- ▶ *Virtualization with IBM Workload Deployer: Designing and Deploying Virtual Systems*, SG24-7967

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at this website:

<http://www.ibm.com/redbooks>

## Online resources

The following websites also are relevant as further information sources:

- ▶ IBM PureSystems Centre:  
<http://www-01.ibm.com/software/brandcatalog/puresystems/centre/>
- ▶ How to use pattern editor:  
[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.webapp.doc%2Fap%2Fapc\\_prodreqs.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.webapp.doc%2Fap%2Fapc_prodreqs.html)
- ▶ IBM BPM Advanced Pattern requirements:  
[http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp?topic=/com.ibm.wbpm.cloud.doc/topics/cbpm\\_priclo\\_gsg.html](http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp?topic=/com.ibm.wbpm.cloud.doc/topics/cbpm_priclo_gsg.html)
- ▶ IBM Mobile Application Platform Pattern:  
[http://pic.dhe.ibm.com/infocenter/wrklight/v5r0m6/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Fc\\_pureapp\\_installation.html](http://pic.dhe.ibm.com/infocenter/wrklight/v5r0m6/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Fc_pureapp_installation.html)
- ▶ Installing IBM Mobile Application Platform Pattern:  
[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Ft\\_pureapp\\_creating\\_mobile\\_application\\_platform\\_pattern.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html)

- ▶ Building an IBM Worklight virtual application:  
[http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Ft\\_pureapp\\_creating\\_mobile\\_application\\_platform\\_pattern.html](http://pic.dhe.ibm.com/infocenter/wrklight/v6r0m0/index.jsp?topic=%2Fcom.ibm.wrklight.help.doc%2Fpureapp%2Ft_pureapp_creating_mobile_application_platform_pattern.html)
- ▶ Creating plug-ins for virtual application patterns, Part 1: An introduction:  
[http://www.ibm.com/developerworks/websphere/library/techarticles/1212\\_dejesus2/1212\\_dejesus2.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1212_dejesus2/1212_dejesus2.html)
- ▶ Plug-in development guide:  
[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2Fwd%2Fpgt\\_pluginovw.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/index.jsp?topic=%2Fcom.ibm.worlodep.doc%2Fwd%2Fpgt_pluginovw.html)
- ▶ Pattern types:  
[http://www.ibm.com/developerworks/websphere/library/techarticles/1204\\_brown/1204\\_brown.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1204_brown/1204_brown.html)
- ▶ Best practices for patterns adoption in IBM PureApplication System:  
[http://www.ibm.com/developerworks/websphere/techjournal/1307\\_brown1/1307\\_brown1.html](http://www.ibm.com/developerworks/websphere/techjournal/1307_brown1/1307_brown1.html)
- ▶ How PureApplication System can affect the existing roles  
[http://www.ibm.com/developerworks/cloud/library/cl-ps-aim1305\\_alignorg/](http://www.ibm.com/developerworks/cloud/library/cl-ps-aim1305_alignorg/)
- ▶ BM SmartCloud Orchestrator installation and configuration:  
[http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc\\_2.2/welcome.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc_2.2/welcome.html)
- ▶ Description of script packages:  
[http://www.ibm.com/developerworks/websphere/techjournal/0911\\_stelzer/0911\\_stelzer.html](http://www.ibm.com/developerworks/websphere/techjournal/0911_stelzer/0911_stelzer.html)
- ▶ How to create your own generic fix:  
[http://www.ibm.com/developerworks/websphere/techjournal/1001\\_amrhein/1001\\_amrhein.html](http://www.ibm.com/developerworks/websphere/techjournal/1001_amrhein/1001_amrhein.html)
- ▶ IBM SmartCloud Orchestrator information center:  
[http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc\\_2.2/welcome.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc_2.2/welcome.html)
- ▶ TOSCA pattern type in SCO:  
[http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc\\_2.2/t\\_to\\_sca\\_enabling.html](http://pic.dhe.ibm.com/infocenter/tivihelp/v48r1/topic/com.ibm.sco.doc_2.2/t_to_sca_enabling.html)
- ▶ Information about TOSCA;  
[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=tosca](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca)
- ▶ IBM PureApplication System W1500 Information Center:
  - ▶ [http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.pure.systems.appsyst.1500.doc%2Fwd%2Fapc\\_monitoring.htm](http://pic.dhe.ibm.com/infocenter/psappsys/v1r1m0/index.jsp?topic=%2Fcom.ibm.pure.systems.appsyst.1500.doc%2Fwd%2Fapc_monitoring.htm)
- ▶ IBM Tivoli Monitoring shared service:  
<https://www.ibm.com/developerworks/community/files/basic/anonymous/api/library/b6f53616-38e7-47ca-bc35-4cc465e5fcb7/document/dd431fb0-f8e1-43be-b0d4-b035755dd097/media>

- IBM Workload Deployer:

[http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/cct\\_usingcli.html](http://pic.dhe.ibm.com/infocenter/worlodep/v3r1m0/topic/com.ibm.worlodep.doc/iwd/cct_usingcli.html)

## Help from IBM

IBM Support and downloads

<http://www.ibm.com/support>

IBM Global Services

<http://www.ibm.com/services>









# Cloud Computing Patterns of Expertise



## Discover pattern portability

## Understand pattern lifecycles

## Build repeatable patterns

This IBM Redpaper publication explains the business and technical value of emerging patterns of expertise in cloud computing, with specific applicability to IBM PureApplication System, IBM Workload Deployer, IBM SmartCloud Orchestrator, and IBM SmartCloud Application Services. It explains how patterns help companies use the different cloud environments that IBM offers. Also included are some preferred practices for helping to ensure pattern portability.

The pattern-based approach is a response to the need to reduce complexity in IT environments, where various skills are required to design, test, configure, and maintain integrated solutions, including clouds. IT managers spend most of their time maintaining applications and application environments, leaving little time to focus on new business needs or to adopt new technologies. As a result, businesses can lack the agility that is needed to be successful in fast-paced, competitive markets.

Pattern of expertise are designed to deliver the following benefits:

- ▶ Faster time-to-value
- ▶ Reduced costs and resource demands
- ▶ Fewer errors and, therefore, lower risk

Patterns make full use of the unique nature of clouds, both private or public. When they are used in the cloud, patterns allow for the dynamic and efficient use of IT resources to achieve consistent results, even when complex solutions are built. In this way, patterns help save time, money, and resources.

This Redpaper aims to show the value that patterns bring to IT managers and the business as a whole.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)