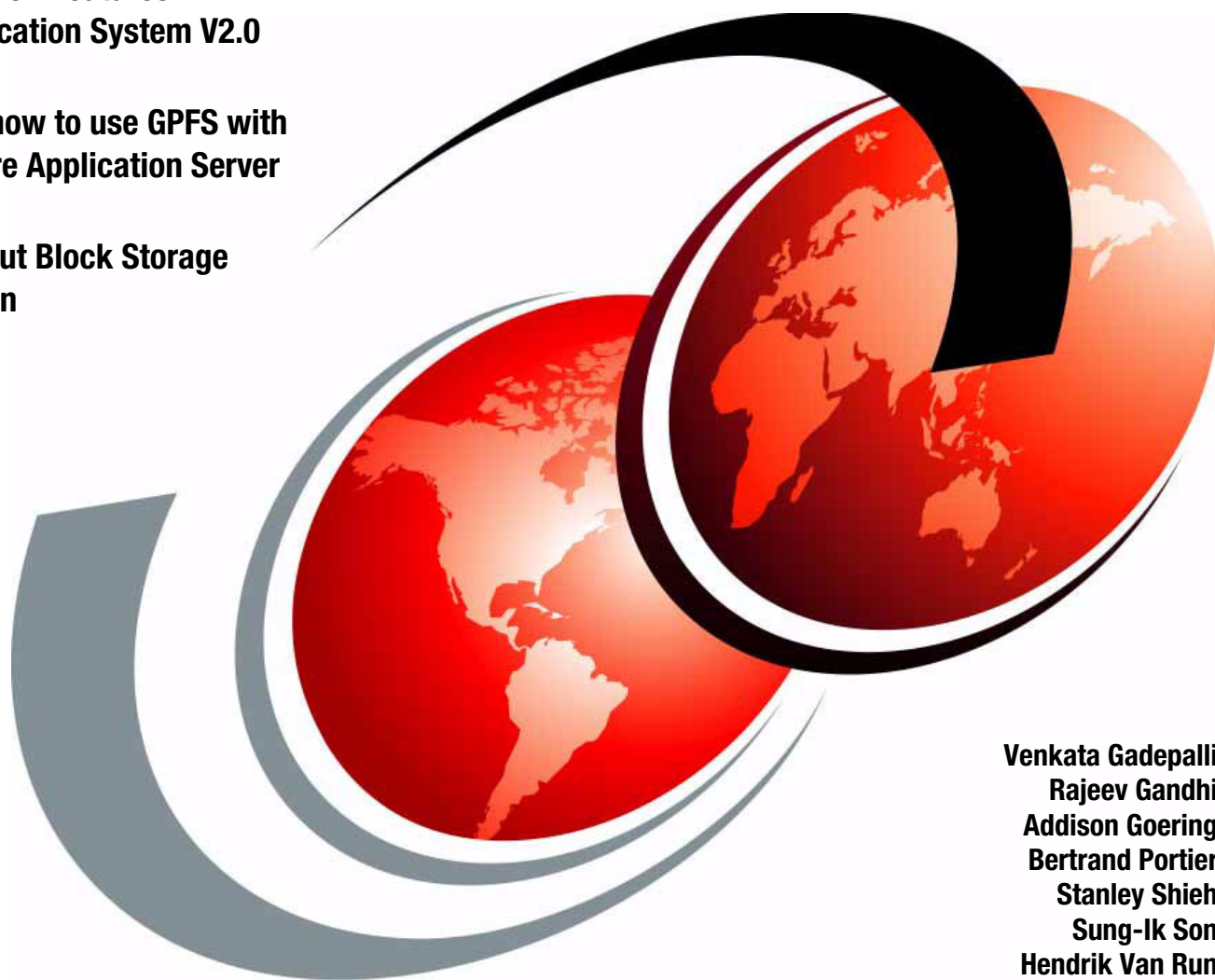


Implementing High Availability and Disaster Recovery in IBM PureApplication Systems V2

Discover new features in IBM PureApplication System V2.0

Examine how to use GPFS with WebSphere Application Server

Learn about Block Storage Replication



Venkata Gadepalli
Rajeev Gandhi
Addison Goering
Bertrand Portier
Stanley Shieh
Sung-Ik Son
Hendrik Van Run

Redbooks



International Technical Support Organization

**Implementing High Availability and Disaster Recovery
in IBM PureApplication Systems V2**

January 2015

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

First Edition (January 2015)

This edition applies to Version 2, IBM PureApplication System.

© Copyright International Business Machines Corporation 2015. 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

| | |
|---|------|
| Notices | ix |
| Trademarks | x |
| IBM Redbooks promotions | xi |
| Preface | xiii |
| Authors | xiii |
| Now you can become a published author, too! | xvi |
| Comments welcome | xvi |
| Stay connected to IBM Redbooks | xvii |
| Chapter 1. Overview | 1 |
| 1.1 Define high availability and disaster recovery | 2 |
| 1.1.1 High availability | 2 |
| 1.1.2 Disaster recovery | 3 |
| 1.2 PureApplication System support for HA and DR | 5 |
| 1.2.1 IBM General Parallel File System (GPFS) | 5 |
| 1.2.2 Shared service for GPFS | 6 |
| 1.2.3 Block storage | 6 |
| 1.2.4 Block storage replication | 7 |
| 1.2.5 External storage | 7 |
| 1.2.6 Multisystem deployments | 8 |
| 1.3 Backup and recovery | 8 |
| 1.4 Always On (Continuous Availability) | 9 |
| 1.4.1 Always On Principles (Continuous Availability) | 9 |
| 1.4.2 Always On Patterns | 10 |
| 1.5 Overview of HADR use case scenarios | 12 |
| Chapter 2. High availability and disaster recovery capabilities of PureApplication System V2.0 | 17 |
| 2.1 Storage volumes | 18 |
| 2.1.1 The new type of storage volumes in PureApplication System V2.0 | 18 |
| 2.2 Block storage overview | 18 |
| 2.2.1 Block storage replication | 19 |
| 2.3 Block storage replication | 20 |
| 2.3.1 Planned failover | 22 |
| 2.3.2 Unplanned failover | 22 |
| 2.4 External storage | 23 |
| 2.5 GPFS Overview | 24 |
| 2.6 GPFS topologies | 25 |
| 2.7 Active/Active GPFS deployment | 25 |
| 2.8 Active/Passive GPFS deployment | 27 |
| 2.9 Shared Service for GPFS | 27 |
| 2.10 GPFS file systems and file sets | 28 |
| 2.11 Load balancing | 28 |
| Chapter 3. High availability and disaster recovery scenarios | 29 |
| 3.1 Overview for the scenarios | 30 |
| 3.1.1 Nomenclature | 30 |

| | | |
|-------|---|-----------|
| 3.1.2 | Patterns | 30 |
| 3.1.3 | Rack topology | 30 |
| 3.1.4 | PureApplication Platform for testing scenarios | 31 |
| 3.1.5 | Scenario basics | 31 |
| 3.2 | HADR scenarios for WebSphere Application Server | 31 |
| 3.2.1 | Scenario WAS_1: WebSphere cell in the same rack (PDC-1) with transactions in GPFS | 32 |
| 3.2.2 | Scenario WAS_2: WebSphere cell across two racks in same data center | 33 |
| 3.2.3 | Scenario WAS_3: WebSphere active-passive cells - identical setup in PDC and SDC, with WebSphere transactions stored in GPFS | 34 |
| 3.3 | HADR scenarios for DB2 | 35 |
| 3.3.1 | Scenario DB2_1: DB2 HADR from the same pattern and deployed on a single rack (PDC-1). | 36 |
| 3.3.2 | Scenario DB2_2: DB2 HADR from the same pattern and deployed the parts on two racks (PDC-1 and PDC-2) | 37 |
| 3.3.3 | Scenario DB2_3: Identical DB2 HADR deployments across primary (PDC) and secondary DR (SDC) data centers | 38 |
| 3.4 | HADR scenarios for WebSphere Application Server and DB2 | 39 |
| 3.4.1 | Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database | 40 |
| 3.4.2 | Scenario WDB_2: WebSphere Application Server cluster split across two racks in the same data center with DB2 HADR also split across the racks, with WebSphere transactions stored in database | 41 |
| 3.4.3 | Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in DB | 42 |
| 3.5 | HADR scenarios for WebSphere MQ | 43 |
| 3.5.1 | Scenario WMQ_1: WebSphere MQ primary and standby in the same pattern deployed on a single rack. | 44 |
| 3.5.2 | Scenario WMQ_2: WebSphere MQ primary and standby in different pattern deployed on two different racks within the same data center. | 45 |
| 3.5.3 | Scenario WMQ_3: WebSphere MQ primary and passive in the different patterns deployed on separate racks across the data center. | 46 |
| | Chapter 4. Infrastructure setup | 47 |
| 4.1 | Block storage configuration | 48 |
| 4.1.1 | Block storage configuration | 48 |
| 4.2 | Block storage replication configuration | 49 |
| 4.2.1 | Block storage replication: Steps | 49 |
| 4.3 | Configuring an Active/Active (Mirrored) GPFS deployment | 53 |
| 4.3.1 | Active/Active GPFS deployment: Steps | 53 |
| 4.4 | Configuring an Active/Passive GPFS deployment | 61 |
| 4.4.1 | Active/Passive GPFS deployment: steps | 61 |
| 4.4.2 | Active/Passive setup and takeover | 62 |
| 4.5 | Deploy GPFS Shared Service | 63 |
| 4.6 | External storage configuration | 64 |
| 4.7 | Network configuration and cloud resources configuration | 65 |
| 4.7.1 | Network configuration | 65 |
| 4.7.2 | Cloud resources configuration | 66 |
| 4.8 | Multisystem environment deployment | 71 |
| 4.8.1 | Management domains | 72 |
| 4.8.2 | Deployment subdomains | 72 |
| 4.8.3 | Additional requirements | 73 |
| 4.8.4 | System configuration | 73 |

| | | |
|--|--|------------|
| 4.8.5 | Create one or more deployment subdomains | 76 |
| 4.8.6 | Cloud resources configuration | 78 |
| 4.9 | DNS setup for primary and secondary (cross) rack scenarios | 81 |
| 4.9.1 | Network setup | 82 |
| 4.10 | Network configuration for WebSphere Application Server and DB2 scenarios | 85 |
| 4.10.1 | PDC | 86 |
| 4.10.2 | SDC | 87 |
| Chapter 5. High availability and disaster recovery scenarios for DB2 | | 89 |
| 5.1 | Introduction to DB2 HADR | 91 |
| 5.1.1 | Primary, standby, and log shipping | 91 |
| 5.1.2 | DB2 client and automatic client rerouting | 91 |
| 5.2 | DB2 Client Setup | 92 |
| 5.2.1 | Deploy DB2 client on PureApplication System | 92 |
| 5.2.2 | Configure DB2 client | 93 |
| 5.3 | Building a DB2 Virtual System Pattern | 93 |
| 5.3.1 | Cloning the Default DB2 OLTP Pattern with HADR for Linux | 94 |
| 5.3.2 | Modifying the new pattern, ITSO DB2 OLTP HADR Pattern | 95 |
| 5.3.3 | Optional: Multiple HADR databases | 99 |
| 5.4 | Scenario DB2_1 | 109 |
| 5.4.1 | Deployment | 110 |
| 5.4.2 | Validation | 113 |
| 5.5 | Scenario DB2_2: Two systems using a single pattern | 114 |
| 5.5.1 | Deployment | 115 |
| 5.5.2 | Validation | 119 |
| 5.6 | Scenario DB2_3: Two systems using block storage replication | 119 |
| 5.6.1 | Adding block storage to Virtual System Pattern | 120 |
| 5.6.2 | Block Storage configuration | 124 |
| 5.6.3 | Deploy Virtual System Pattern on both systems | 127 |
| 5.6.4 | Enable Block Storage replication | 132 |
| 5.6.5 | Validate the Virtual System Instance on the active system | 142 |
| 5.6.6 | Planned Failover to SDC | 142 |
| 5.6.7 | Unplanned Failover to SDC | 156 |
| 5.7 | Validation | 160 |
| 5.7.1 | Configure the DB2 client | 160 |
| 5.7.2 | Connect to the DB2 database and perform a simple query | 162 |
| 5.7.3 | Confirm DB2 HADR roles of primary and standby DB2 servers | 163 |
| 5.8 | Testing for outages | 165 |
| 5.8.1 | Planned outage: DB2 takeover | 165 |
| 5.8.2 | Unplanned outage - shutdown of primary database OS | 168 |
| Chapter 6. High availability and disaster recovery scenarios for WebSphere Application Server | | 173 |
| 6.1 | Scenario WAS_1: WebSphere cell in the same rack, transactions in GPFS | 174 |
| 6.1.1 | Configure Primary GPFS Server | 174 |
| 6.1.2 | Build the WebSphere Application Server cluster pattern | 175 |
| 6.1.3 | Add GPFS Client Policy | 175 |
| 6.1.4 | Deploy WebSphere Application Server cluster pattern | 177 |
| 6.1.5 | Create a WebSphere Application Server cluster | 178 |
| 6.1.6 | Configure transaction services | 178 |
| 6.1.7 | Test Scenario WAS_1 HA | 178 |
| 6.2 | Scenario WAS_2: Single WebSphere Cell Across two racks in PDC | 179 |
| 6.2.1 | Configure and Deploy GPFS Mirror Server at PDC-2 | 180 |
| 6.2.2 | Configure and Deploy Tiebreaker Server at PDC-2 | 182 |

| | | |
|--|--|-----|
| 6.2.3 | Configure and deploy GPFS Primary server on PDC-1 | 183 |
| 6.2.4 | Deploy GPFS Shared Service at PDC-1 | 185 |
| 6.2.5 | Configure WebSphere Application Server with GPFS client policy | 186 |
| 6.2.6 | Deploy WebSphere pattern to multiple domains | 186 |
| 6.2.7 | Configure WebSphere Application Server to write transaction log to GPFS storage volume | 187 |
| 6.2.8 | Test the Multi-domain WebSphere Split Cell HA using GPFS scenario | 188 |
| 6.3 | Scenario WAS_3: Active/Passive, identical setups in PDC and SDC, with transactions stored in GPFS | 189 |
| 6.3.1 | Network configuration | 191 |
| 6.3.2 | Cloud Group and Environment Profile | 191 |
| 6.3.3 | Storage Volume | 192 |
| 6.3.4 | GPFS | 193 |
| 6.3.5 | WebSphere Application Server | 193 |
| 6.3.6 | Planned outage at PDC | 198 |
| 6.3.7 | Recovery | 199 |
| | | |
| Chapter 7. High availability and disaster recovery scenarios for WebSphere Application Server and DB2 | | |
| 7.1 | Common assets used in scenarios | 206 |
| 7.2 | Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database | 207 |
| 7.2.1 | Build the WebSphere Application Server cluster pattern | 207 |
| 7.2.2 | Deploy WebSphere Application Server cluster pattern | 207 |
| 7.2.3 | Create a WebSphere Application Server cluster | 207 |
| 7.2.4 | Install DB2 JDBC driver | 208 |
| 7.2.5 | Create a JDBC Provider | 208 |
| 7.2.6 | Create a J2C alias | 208 |
| 7.2.7 | Create data sources | 208 |
| 7.2.8 | Install BankTransaction application | 208 |
| 7.2.9 | Validate the functionality of the application | 208 |
| 7.3 | Scenario WDB_2: WebSphere Application Server cluster with DB2 HADR, split across two racks with WebSphere transactions stored in database | 209 |
| 7.4 | Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in database | 211 |
| | | |
| Chapter 8. High availability and disaster recovery scenarios for WebSphere MQ | | |
| 8.1 | Common assets used in WebSphere MQ scenarios | 225 |
| 8.1.1 | Image Parts | 225 |
| 8.1.2 | Policies | 226 |
| 8.1.3 | Script packages and parameters used in the pattern | 226 |
| 8.1.4 | Prerequisites | 228 |
| 8.2 | Scenario WMQ_1: WebSphere MQ Primary and standby in the same pattern deployed on a single rack | 228 |
| 8.3 | Scenario WMQ_2: WebSphere MQ primary and standby in different patterns deployed on two different racks in the same data center | 235 |
| 8.4 | Scenario WMQ_3: WebSphere MQ primary and standby in the different patterns deployed on two different racks across different data centers | 241 |
| 8.4.1 | Steps for creating a WebSphere MQ active/passive scenario | 243 |
| | | |
| Appendix A. Sample Application | | |
| | Database Setup | 255 |
| | Create CHECKING database and SAVINGS database | 255 |
| | List database directory | 255 |

| | |
|--|------------|
| Create tables | 256 |
| Insert a single row into the SAVINGS table | 257 |
| Terminate and configure CHECKING database and table | 257 |
| Application Coding Explanation | 258 |
| First phase (withdraw) | 259 |
| Second phase (deposit) | 260 |
| Testing | 260 |
| Appendix B. Common WebSphere Application Server configuration tasks | 265 |
| Build the WebSphere Application Server cluster pattern | 266 |
| Deploy WebSphere Application Server cluster pattern: Single rack | 269 |
| Deploy WebSphere Application Server cluster pattern: Multiple rack | 270 |
| Create WebSphere Application Server cluster. | 273 |
| Configure database connectivity for BankTransaction application. | 273 |
| DB2 JDBC Database driver installation | 274 |
| Create a JDBC Provider | 275 |
| Create a J2C alias | 278 |
| Create data sources for SAVINGS, CHECKING, and transaction data source | 280 |
| Create data source for SAVINGS | 280 |
| Install BankTransaction application | 284 |
| Validate BankTransaction application | 284 |
| Appendix C. Additional material | 287 |
| Locating the web material | 287 |
| Using the web material. | 287 |
| Downloading and extracting the web material | 288 |
| Related publications | 289 |
| IBM Redbooks | 289 |
| Online resources | 289 |
| Help from IBM | 290 |

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](http://www.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® | Power Systems™ | System/390® |
| DB2® | PowerVM® | Tivoli® |
| Everyplace® | PureApplication® | VisualAge® |
| GPFS™ | Redbooks® | WebSphere® |
| IBM® | Redbooks (logo)  ® | |

The following terms are trademarks of other companies:

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

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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.

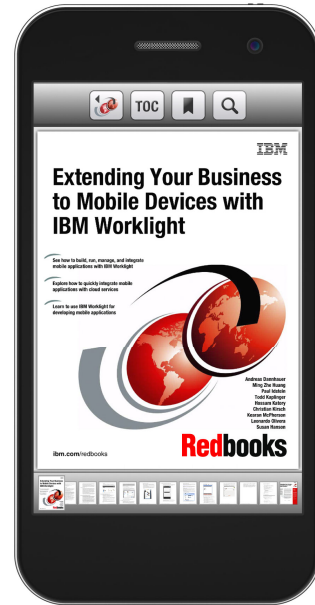
UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication describes and demonstrates common, prescriptive scenarios for setting up disaster recovery for common workloads using IBM WebSphere® Application Server, IBM DB2®, and WebSphere MQ between two IBM PureApplication® System racks using the features in PureApplication System V2.

The intended audience for this book is pattern developers and operations team members who are setting up production systems using software patterns from IBM that must be highly available or able to recover from a disaster (defined as the complete loss of a data center).

Authors

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



Venkata Gadepalli is a member of the IBM Software Services for WebSphere team. He has more than 16 years of experience in the IT field and has been involved in client engagements involving the WebSphere family of products. His main area of interest is working with first-of-a-kind engagements that involve IBM PureApplication System. Vishy has written numerous papers that have been published both within and outside of IBM. He also co-authored the first and second editions of the IBM WebSphere Portal Primer (IBM Press, 2003 and 2005).



Rajeev Gandhi is a Senior Technical Staff Member in the IBM Software Services for WebSphere team. He has more than 28 years of experience in the IT field and has been involved in client engagements working with many of the early releases of IBM middleware products. His current main area of focus is IBM PureApplication System. He was the lead developer on the enablement team for IBM PureApplication System, and currently works with clients to help them adopt IBM PureApplication Systems. He has presented in several IBM and external conferences, and written articles in DeveloperWorks. Rajeev holds a M.S. in Computer Science from University of Connecticut.



Addison Goering is a Certified IT Specialist with the WebSphere Education team. His main specialty is the design, development, and delivery of courses in the WebSphere product family. He has developed and delivered courses ranging from webinars to week-long workshops on products such as WebSphere ESB, IBM Workload Deployer, WebSphere Application Server, WebSphere Business Services Fabric, and IBM WebSphere BPM. He is the lead developer on the WebSphere Education team that is developing education on IBM PureApplication System. Addison holds a B.S. in education from Keene State College in New Hampshire, mainframe certification from DePaul University in Chicago, and several certifications from IBM.



Bertrand Portier is an IBM Executive IT Architect in the United States. He has 15 years of experience in the field of distributed computing. He holds a Masters degree in Computer Engineering from Lille Polytechnique School. His areas of expertise include IT Architecture, technical leadership, innovation, and Always On. He has written extensively on these topics.



Stanley Shieh is an Executive IT Specialist. He started his IBM career learning the assembly language for the mainframe. The last language that he learned from work was the Smalltalk. He left his development work after 10 years as a developer. He went to the field as a Java advocate and later became a technical sales engineer and manager. In his field days, he supported IBM VisualAge® for Smalltalk, VisualAge for Java, WebSphere Application Server, Edge Server, Portal Server, WebSphere MQ and BPM. His current focus is on PureApplication System. He supports IBM North America. He has degrees in Industrial Management Science, Management Information System, and Computer Science. He co-authored *Connecting the Enterprise to the Internet with MQSeries and VisualAge for Java*, SG24-2144, *Application Development with VisualAge for Smalltalk and MQSeries*, SQ24-2117, and *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition*, SG24-6153.



Sung-Ik Son is a Senior Software Engineer at IBM in Research Triangle Park, NC, United States. He has been a key developer in IBM system and application software development organizations. He has worked in VM/CP ESA Development, IBM System/390® Client/Server Development, Personal Communication Development, and WebSphere Performance Pack Development. After his successful software development career for 12 years, he moved to the Software Services organization and has worked with various WebSphere products that include WebSphere Application Server, WebSphere Transcoding Publisher, IBM SWebSphere Everyplace® products, WebSphere RFID (Radio Frequency Identification), WebSphere Portal, WebSphere Process Server, WebSphere DataPower, and PureApplication System. Sung-Ik's current focus and area of responsibility is IBM PureApplication System. He holds a B.S. in Computer Science from the University of New Brunswick in Canada and an MS in Computer Science from Purdue University.



Hendrik Van Run is an Executive IT Specialist with IBM Software Services for WebSphere (ISSW) consulting practice in Hursley. He is the European technical lead for PureApplication System and helps clients evaluate and implement solutions using this platform. He has advised many clients on issues such as high availability, performance, and scalability across the WebSphere Application Server product families. Hendrik also has a strong background in performance analysis and optimization within the enterprise. He holds a Master's degree in physics, is co-author of a number of IBM Redbooks, and frequently speaks at conferences.

This project was led by:

Margaret Ticknor 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:

- ▶ Tom Alcott, Senior Technical Staff Member, World Wide WebSphere Application Infrastructure.
- ▶ Kyle Brown, Distinguished Engineer, CTO Cloud and Emerging Technologies, Master Inventor, for his leadership, mentoring, guidance, and inspiration throughout the entire effort of developing this publication. Kyle was involved in the design of the scenarios, review of the book and providing overall guidance throughout this work. Without Kyle's effort and help, the authors would not have been able to complete this work.
- ▶ Arunava Majumdar, Application Integration and Middleware Solutions Specialist: AIM.WS WebSphere Message Broker Architecture, for providing some of the WebSphere MQ script packages used in the WebSphere MQ scenarios.
- ▶ Scott Moonen, Senior Software Engineer, PureApplication System, for his timely and prompt help in resolving issues with the multi-rack setup and networking across the racks.
- ▶ Dan Mullen, Software Developer, IBM PowerVM® Cloud Development, for his timely and prompt help in resolving issues with the multi-rack setup and networking across the racks.

- ▶ Jim Robbins, STSM and lead developer for IBM PureApplication System for providing infrastructure help in setting the PureApplication Systems and managing the systems used to validate the scenarios, helping answer key architecture questions related to scenarios and the new function.
- ▶ Herbie Pearthree, STSM Continuous Availability Svcs., Acting CTO.
- ▶ Valentina Popescu, lead developer for the GPFS™ component in IBM PureApplication System for providing help in answering many questions related to GPFS, which was the key component of the HADR scenarios.
- ▶ Al Weiner, lead developer for WebSphere Application Server Hypervisor components in IBM PureApplication System for answering many questions.

Thanks to the following people for their support of this project:

- ▶ Deana Coble, IBM Redbooks Technical Writer and Video
- ▶ Tamikia Lee, IBM Redbooks Residency Administrator
- ▶ Thomas Edison, IBM Redbooks Graphics Editor
- ▶ Ernest A. Keenan, IBM Redbooks Editor

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:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

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>



Overview

Most carpenters have a toolbox full of tools to use on construction projects. Given a stack of building materials for project, plan blueprints, and knowledge, a carpenter can complete a construction project according to the blueprints. A smart carpenter knows when, where, and how to use those tools.

IBM PureApplication System provides many capabilities that allow you to implement high availability (HA) strategies and disaster recovery (DR) scenarios for your workloads. These capabilities are analogous to the tools in the carpenter's toolbox. You might consider PureApplication System as a toolbox that provides you with the tools to implement HA strategies and DR scenarios. The goal of this book is to introduce you to applying the tools in the toolbox to a number of common scenarios.

Each organization has its own specific high availability disaster recovery (HADR) requirements (plan blueprints). Each organization's requirements are uniquely different from other organizations. Because PureApplication System does not provide a single solution or a cookbook for implementing HA strategies and DR scenarios, the organization must draw upon the collective experience of its staff and the tools in the PureApplication Server toolbox.

This chapter introduces the tools and scenarios in the following topic sections:

- ▶ Define high availability and disaster recovery
- ▶ PureApplication System support for HA and DR
- ▶ Backup and recovery
- ▶ Always On (Continuous Availability)
- ▶ Overview of HADR use case scenarios

1.1 Define high availability and disaster recovery

High availability and disaster recovery are often discussed together. This section begins with some definitions of the concepts.

1.1.1 High availability

If an organization strives for durability and uninterrupted operation without failure, it desires a highly available system. With highly available systems, you might experience outages for a few seconds or even a few minutes while failover occurs. High availability does allow for planned outages, such as system upgrades, but does not allow unplanned outages.

High availability must be considered on the hardware side and on the workload side. The next section provides an overview of hardware and workload high availability characteristics of PureApplication System.

PureApplication System hardware high availability

PureApplication System is designed for HA in its hardware redundancy. Compute nodes, network controllers, management nodes, virtualization nodes, storage controllers, and storage are all redundant and contribute to a highly available environment as shown in Figure 1-1 on page 3. The following list notes the component level details available on a single rack system:

- ▶ **Compute nodes:** The management system automatically routes around failed cores. If an entire node fails, the system tries to move the VM to another compute node in its cloud group if free space is available.
- ▶ **Network controllers:** The cabling and switches are redundant. The failure of one controller reduces bandwidth. Service is continuous.
- ▶ **Management nodes:** The node has a backup server. A floating IP address is assigned to the active management node (workload deployer).
- ▶ **Virtualization nodes:** The node has a backup server.
- ▶ **Storage controllers:** Each controller has two canisters that service all traffic. If one fails, the other handles all traffic.
- ▶ **Storage:** SSD and HDD storage is configured in RAID5 plus spares. Storage is designed to tolerate two concurrent failures without data loss (after the spares are in use).

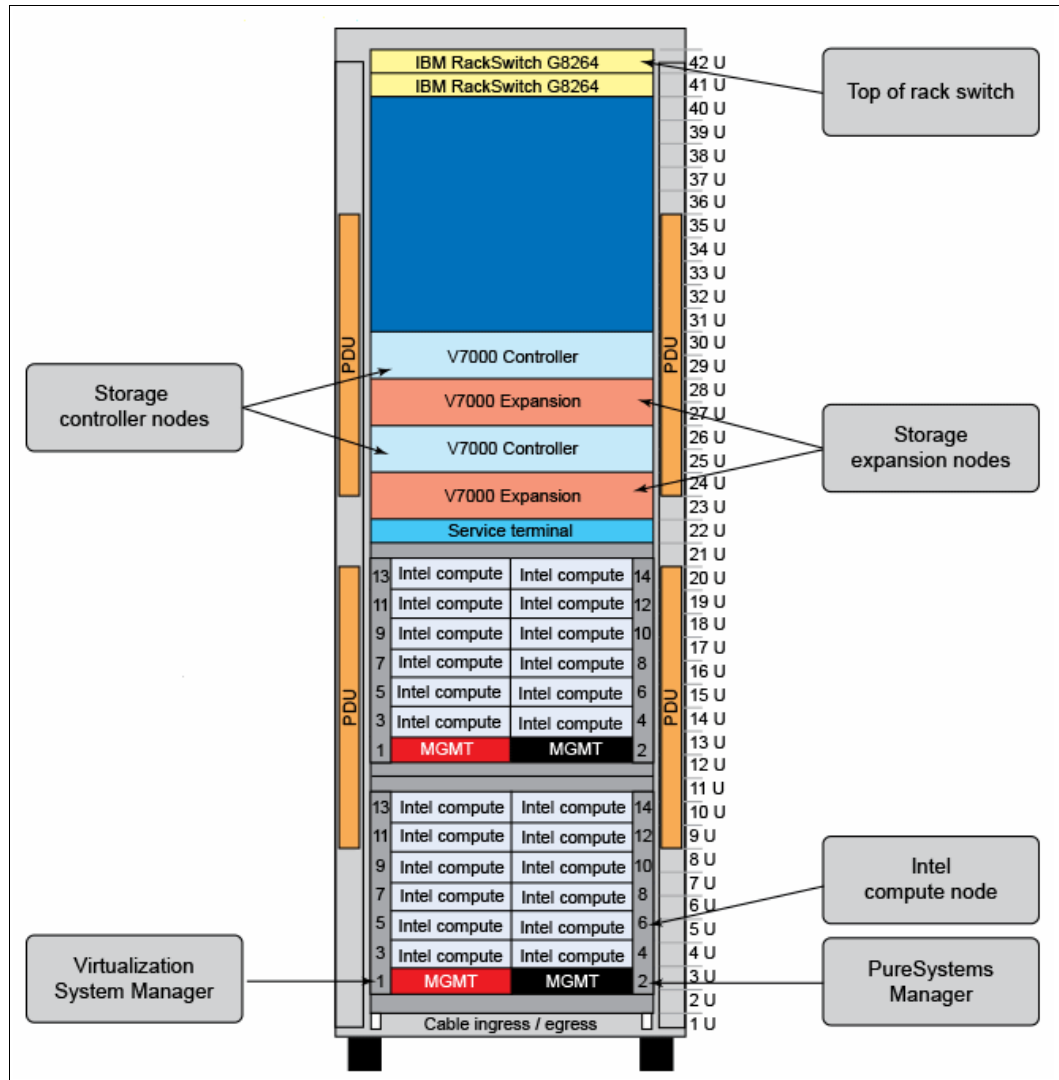


Figure 1-1 PureApplication System - HA through hardware redundancy (model shown W2500-384)

PureApplication System workload high availability

Several of the IBM software products that are implemented as patterns for PureApplication System have high availability characteristics built into them. These patterns include those noted in the following list:

- ▶ DB2 HADR pattern
- ▶ WebSphere Application Server cell composition with deployment manager and built-in script packages
- ▶ Load balancers, such as the WebSphere Application Server On Demand Router and DataPower XI52 Virtual Appliance.

1.1.2 Disaster recovery

Disaster recovery is generally considered a plan or process for reconstructing (recovering) data center operations in a different data center. The goal of disaster recovery is business continuity, which drives the requirements for acceptable service levels. The purpose of DR is to return a site to a running condition restored from a last “good” state.

Business continuity might be a more accurate term when describing disaster recovery. The following list notes the two fundamental measurements that drive an organization toward business continuity:

- ▶ Recovery time objective (RTO) answers the question: “If a disaster renders a system unavailable, how much time do you have to make it available again?”. In other words, RTO is the maximum wanted length of time that is allowed between a disaster and the resumption of normal operations and service levels. A business might define and measure its RTO in days or hours for non-critical business systems, but in minutes or seconds for systems critical to the functioning of the business.
- ▶ Recovery point objective (RPO) answers a different question: “If a disaster renders a system unavailable, what is the time frame in which data might be lost?” RPO addresses the maximum acceptable amount of data loss measured in time due to a disaster. RPO is typically measured in seconds or milliseconds. A business might lose inflight transactions for the last second, but no more data loss is acceptable.

There is no exact formula to determine RTO and RPO. Each organization has different expectations that contribute to RTO and RPO requirements. The challenge in this area is balancing those expectations against the technical complexity of enabling disaster recovery. Figure 1-2 illustrates different RTO and RPO expectations.

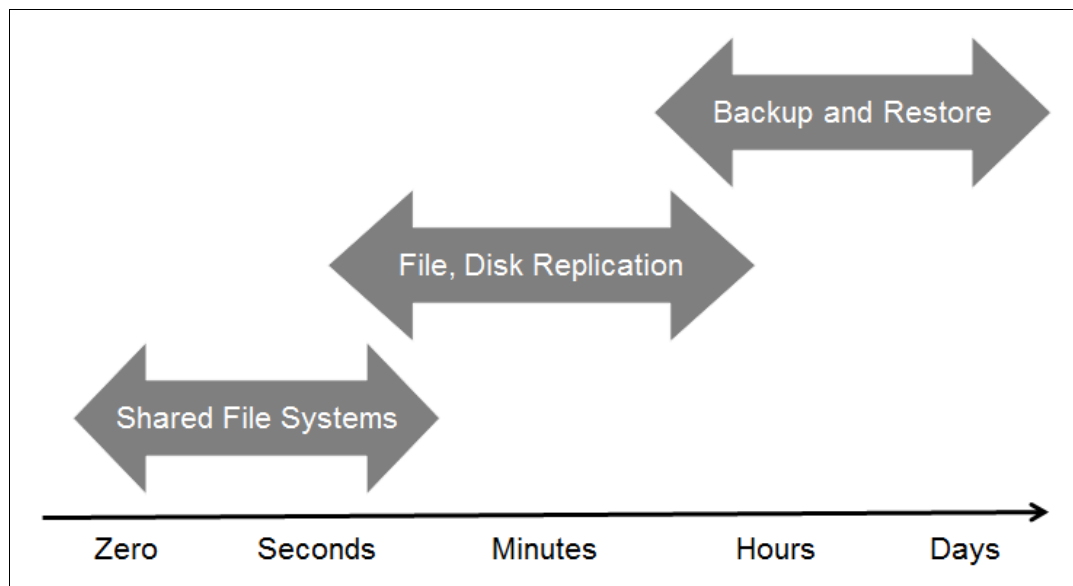


Figure 1-2 Disaster recovery solution ranges

PureApplication System disaster recovery

Three aspects of a PureApplication System must be moved from one system to another to restore functionality of a system’s previous configuration. These aspects are shown in Figure 1-3 on page 5, and noted in the following list:

- ▶ Replicate management data that is stored locally in the storage of the management nodes. This data includes the workload components such as patterns, virtual images, pattern types, script packages, and plug-ins. This data can be synchronized automatically through the multitarget deployment features of PureApplication System V2.0.
- ▶ Replicate application data that include logs, message queues, configuration data, and databases.
- ▶ Redirect network traffic from the primary system to the backup system.

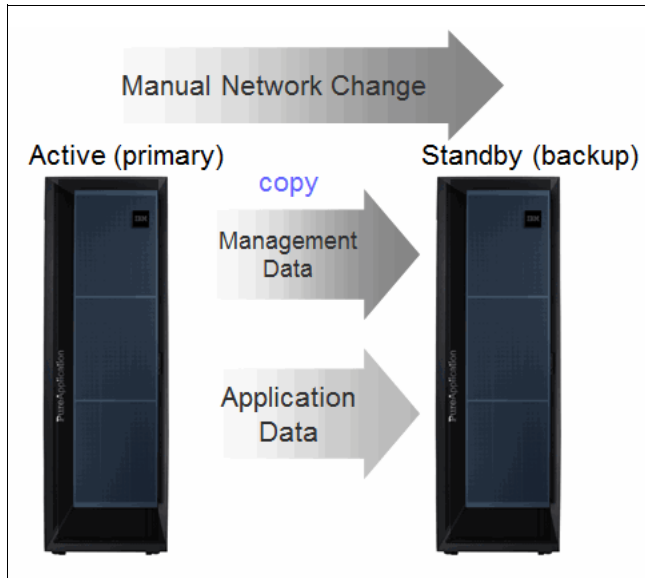


Figure 1-3 PureApplication System disaster recovery basics

1.2 PureApplication System support for HA and DR

This section provides a general overview of the support that is found in IBM PureApplication System V2.0 for high availability and disaster recovery.

1.2.1 IBM General Parallel File System (GPFS)

GPFS is a scalable, high performance file management infrastructure for IBM AIX®, Linux, and Windows Server systems based on a shared disk model, provided by an underlying SAN.

GPFS provides fast, reliable access to data from multiple nodes in a cluster environment. Applications can readily access files using standard file system interfaces, and the same file can be accessed concurrently from multiple nodes. GPFS is designed to provide high availability through advanced clustering technologies, dynamic file system management, and data replication. GPFS can continue to provide data access even when the cluster experiences storage or node malfunctions. GPFS scalability and performance are designed to meet the needs of data intensive applications such as engineering design, digital media, data mining, relational databases, financial analytics, seismic data processing, scientific research, and scalable file serving.

In terms of PureApplication System, GPFS allows multiple nodes or virtual machines to concurrently access the same data. GPFS provides a shared file system to support highly available configurations for virtual system and virtual application pattern deployments within the same rack or across racks.

In the example GPFS configuration shown in Figure 1-4, a GPFS cluster is shown spanning two PureApplication Systems, maximizing the storage within the two systems. The file system is mirrored across the systems. There are other valid GPFS configurations beyond this example.

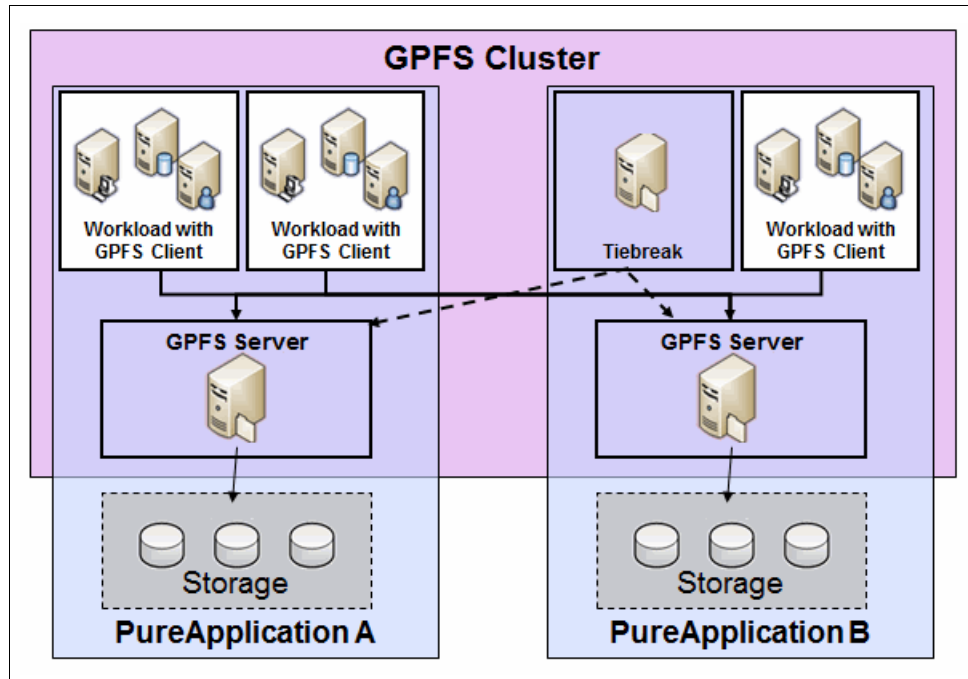


Figure 1-4 An example GPFS configuration

1.2.2 Shared service for GPFS

For workloads to connect to a GPFS server and use shared file systems, the shared service for GPFS must be deployed. The shared service defines the parameters to connect to the GPFS server that connects to shared storage. As a result, any pattern deployed to a cloud group into which a shared service is deployed can use the storage.

In previous versions of PureApplication Server, some client setup was required to use GPFS. To compose a pattern, you added scripts to parts that required access to the GPFS shared file system. There were scripts to share SSH certificates, add a client to GPFS, create a file set, connect to a file set, and link to an existing GPFS file set.

With PureApplication System V2.0, the GPFS Client Policy supports application components in both virtual system patterns and virtual application patterns. When the GPFS Client Policy is added to a pattern component, at deployment time the GPFS product is installed on the virtual machine. Also, the configuration is retrieved from the shared service, and the client is connected to shared file systems that are hosted by the deployed GPFS server pattern.

1.2.3 Block storage

A new type of storage volume, called block storage, was introduced with PureApplication System V2.0. Block storage can be cloned on a single rack or replicated to another rack for HADR solutions. Block storage takes advantage of storage controller LUNs, thus directly avoiding Virtual Machine File System (VMFS) and allowing more capabilities. Volumes can be internal volumes or defined in an external storage device (see 1.2.5, "External storage" on page 7).

Implementing block storage separates the application and data lifecycles. If an application is deleted, block storage is persisted. Block storage is not included in application snapshots. Internal block storage volumes can be exported, imported, and replicated to another rack. Block storage volumes support synchronous replication (0 - 300 kilometers) and asynchronous replication (0 - 8000 kilometers).

1.2.4 Block storage replication

Block storage replication is a form of disaster recovery at the individual storage volume level. Rather than replicating the entire system configuration, you can configure replication of each storage volume between local and remote systems in either direction. As noted previously in 1.2.3, “Block storage” on page 6, replication can occur in synchronous (< 300 km) or asynchronous modes (< 8000 km) depending on your environment. Block storage replication removes the need to have a dedicated disaster recovery rack.

1.2.5 External storage

You can attach your own storage to a PureApplication System rack or even multiple racks as shown in Figure 1-5. This concept is similar to connecting two networks through an Ethernet switch, but you are connecting drives instead of computers. Attaching external volumes to multiple racks allows high availability across PureApplication System hardware.

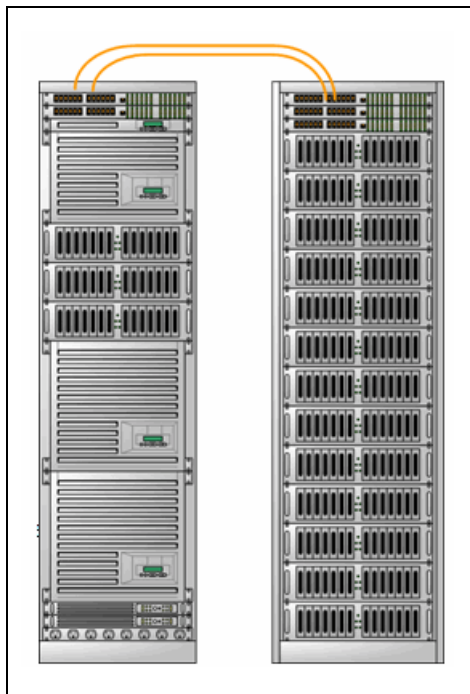


Figure 1-5 External storage

When you use external storage devices, you can assign storage volumes to cloud groups as either block or block shared storage volume types. Block shared storage is available on Intel systemsonly. You provide the external SAN Volume Controller hardware. For a list of supported SAN Volume Controller hardware, see the IBM Knowledge Center at:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extstorage_plan.dita?lang=en

The PureApplication System provides the following listed connections:

- ▶ Ethernet connections for SAN Volume Controller management to PureApplication System rack
- ▶ Fibre Channel connections for customer storage

1.2.6 Multisystem deployments

The multisystem deployment capability is also referred to as *multirack deployment* or *multitarget deployment*. The concept behind multi-system deployments is to improve the availability and resilience of applications by deploying patterns across multiple PureApplication System systems. You can deploy virtual system patterns, virtual application patterns, and shared services to a multisystem environment. At deployment time, the system uses a placement algorithm to determine where the deployment is placed. You can move virtual systems to different cloud groups, but the system first verifies the availability of the necessary resources and artifacts.

Note: Classic virtual system patterns and promoted classic virtual system patterns can be deployed only in a single rack system.

PureApplication System employs the concept of a *management domain* to create and manage the relationship between two or more PureApplication System systems. The management domain is used to manage users and artifacts in multiple systems. There is no limit to the number of systems in the management domain. There is also no limit on the distance between systems in the management domain. Management data can be replicated between systems in a management domain.

A *deployment subdomain* is used to deploy instances from a single pattern across multiple PureApplication System systems. The systems must be part of the same management domain and membership is limited to two systems. Consider the following hardware requirements:

- ▶ The systems must be connected by a low-latency network
- ▶ When there is no connectivity between systems, a 1 GB storage iSCSI target is used as a tiebreaker

1.3 Backup and recovery

Although not discussed in detail in this book, using the PureApplication System capabilities for systems backup and restore provides a simple level of disaster recovery that you can use if your business continuity requirements are lenient, such as an RTO of 48 hours and an RPO of 24 hours. The general approach, which is outlined in the following list, fulfills the disaster recovery requirements shown in Figure 1-3 on page 5. The location defined as the backup must be on a UNIX or Linux server with SSH configured. You can also perform many of the tasks in this approach using REST API commands.

The following list gives an approach outline for disaster recovery requirements:

- ▶ Use one rack for production (active/primary) and another rack for non-production (standby/backup).
- ▶ To replicate management data, take the following actions:
 - On the backup rack, configure a production cloud group for disaster recovery that includes IP groups. The IP groups should contain production IP addresses and VLANs.
 - The backup rack should contain a single compute node.
 - The remainder of the backup rack is used for non-production work.
- ▶ Take application data backups of each pattern instance every 24 hours.
- ▶ To be able to fail over the active rack to the backup rack, take the following actions:
 - On the backup rack:
 - Store or delete workloads in the non-production cloud group to release resources.
 - Quiesce, stop, and remove compute nodes from the non-production cloud group.
 - Add compute nodes to the Prod cloud group.
 - Deploy patterns for the workloads to be recovered.
 - Restore application data backups into the newly deployed workloads.
- ▶ Redirect network traffic to make the workloads available. This action makes the active workloads available on the backup.

1.4 Always On (Continuous Availability)

The need for high levels of availability, *Continuous Availability*, has been driven by the confluence of cloud, analytics, mobile, and social media. Always On or Continuous Availability is expensive and requires no visible planned or unplanned outages.

Always On is not for all applications. There is still a maturity level and a high cost required to achieve Always On. The level of availability for Always On is at or above five 9's, or 99.999% up time (typically with RTO and RPO close to 0). Five 9s means only 27 seconds of downtime per month and requires automated intervention. This type of high availability has traditionally been expected for the top 5 - 10 percent of applications (referred to as tier 0 or tier 1).

Almost Always On (*near* Continuous Availability) refers to slightly lower up times where human intervention is required.

1.4.1 Always On Principles (Continuous Availability)

Always On is based on the following three principles:

- ▶ The ability to withstand component failures:

This is where you can use best practices from high availability concepts and many of the topics described in this book. This book uses techniques such as component redundancy and the elimination of single points of failure. Components can fail at any level of the stack (for example, network layer, systems layer, middleware layer, or application layer). Always On is achieved by implementing component redundancy and clustering. As described in this book, many of these availability functions are pre-built features within the PureApplication System.

- ▶ The ability to withstand catastrophes transparently:

This is designed into your topology with out of region data centers. Out of region means that another data center is on a different power network in a different geographic region. For example, this can involve having PureApplication systems deployed in multiple regions, typically hundreds of miles apart. This is a different approach to disaster recovery. In this approach, you do not wait for the disaster before you are using the infrastructure that is out of region. With the Always On approach, the out of region data center is already up and running and plays an active role in the architecture (Active/Active).
- ▶ The ability to introduce change non-disruptively:

This is related to the domain of Continuous Operations and eliminating planned outages. Customers are used to their phone applications evolving and needing updates to obtain new functionality for services. The ability to introduce innovations and new versions of the applications on a frequent basis falls in to this category. DevOps practices are important to success in this Always On category.

1.4.2 Always On Patterns

Implementing Always On strategies involves designing topologies with out of region (geographically dispersed) architectures using Active/Active settings, combined with techniques for introducing changes (upgrades) without breaking the availability promise. This section describes several Always On application infrastructure patterns (topologies) that can help to achieve those Always On principals.

It is important to note the following aspects before reviewing the patterns:

- ▶ The first Always On application infrastructure pattern, Active/Standby, is not a true Always On strategy. Rather, it is more of a common starting point for Always On.
- ▶ These examples assume that you are on a layered architecture where any specific layer depends on the underlying layers to be Always On. For example, applications need to be implemented with Always On concepts as a part of their criteria. These applications rely on the middleware (application infrastructure), such as WebSphere Application Server. The middleware relies on the virtual images or containers, which rely on the operating system, which relies on the system hardware and networking.
- ▶ Data consistency requirements are going to drive the selection of a specific pattern. Consider, for example, there is latency involved when making dual writes or when replicating data across distances. If you want a response time of 100 ms and you need to do synchronous writes, then the two data centers must be within metro distance (typically no more than 40-100 km apart). When absolute data consistency is required (such as for atomicity, consistency, isolation, durability (ACID) requirement), then only a subset of the patterns can be used. Because of varying requirements, an overall architecture will typically use different patterns for different needs. For example, one pattern for the system of record (absolute data consistency requirement for the core banking system on DB2) and another pattern for the system of engagement (eventual data consistency for the online banking application on WebSphere Application Server).

As mentioned previously, there are several Always On (or Almost Always On) patterns. The following list notes those patterns at a high level (see also Figure 1-6 on page 11):

- ▶ Active/Standby metro with out of region disaster recovery
- ▶ Active/Active metro with out of region disaster recovery
- ▶ Active/Active metro with out of region query
- ▶ Active/Active/Active out of region

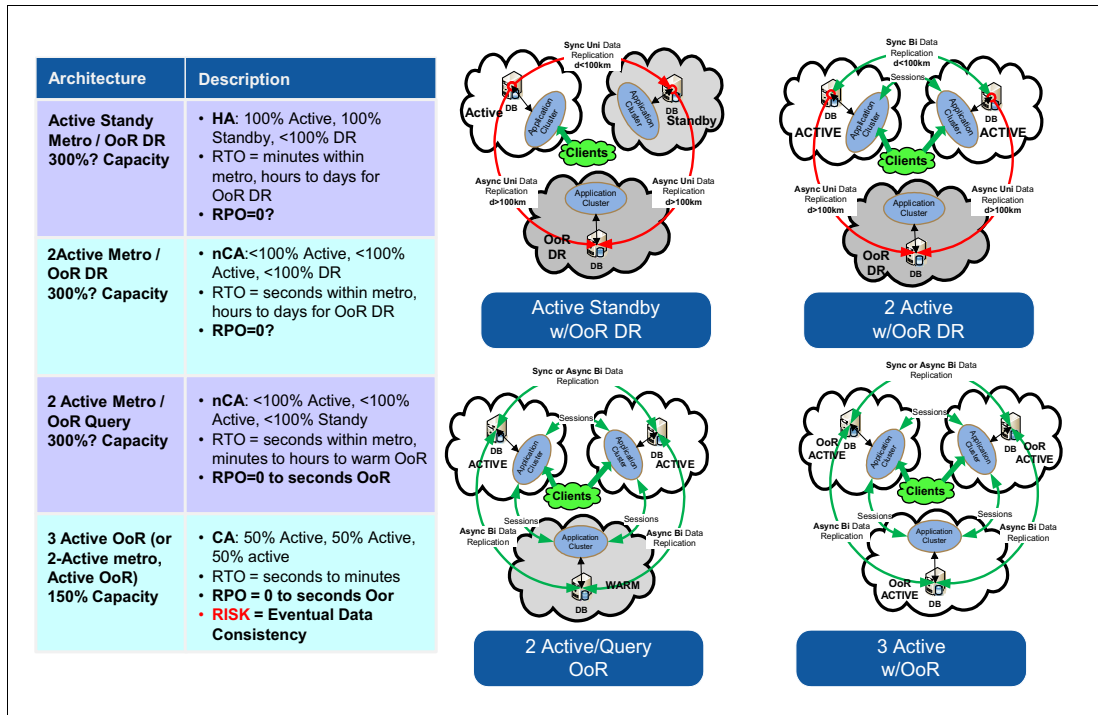


Figure 1-6 High-level view of the various Always On topology patterns

The following list gives more details about these Always On topology patterns:

- ▶ **Active/Standby metro with out of region disaster recovery:** This pattern is the most widely used, and a starting point for Always On. With Active-Standby, one of the data centers is active and accepting client requests. The data is synchronously replicated to the in-metro standby data center, and asynchronously replicated to the out of region data center. If the active data center goes down, traffic is routed to the standby.
- ▶ **Active/Active metro with out of region disaster recovery:** This pattern has two within-metro data centers that accept requests (are active). Synchronous data replication occurs between these two centers and asynchronous data replication to the out-of-region data center. A global routing strategy is in place to route clients to either active data center. If one of the centers fail, the other one takes over.
- ▶ **Active/Active metro with out of region query:** This pattern is a further developed modification of the previous Active/Active pattern. In this pattern, now including out of region query, the out of region data center is *warm* (infrastructure in place and operational with some or most data) and is used for read-only queries. This pattern is the most comprehensive for absolute data consistency (ACID) requirements.
- ▶ **Active/Active/Active:** This pattern has three out-of-region active data centers, all accepting requests. Data is asynchronously replicated to the other centers.

In this section, the Always On topology patterns were presented at a high level. For more information about these topology and patterns (including economical considerations on capacity and rational for multi-site data centers) are available, see *Always On: Assess, Design, Implement, and Manage Continuous Availability*, REDP-5109.

For the PureApplication System use case scenarios described in this book, note the following observations for their use of Always On patterns:

- ▶ **Geographic dispersion:** The PureApplication System use case scenarios, described in this book, are all implemented with one or two PureApplication systems in the same data center. This is because of the test environment available when writing the book. If you were using IBM PureApplication System topologies that span distances, then you would need to pay attention to the constraints associated with latency (the time it takes to replicate) when addressing the data consistency requirements. PureApplication System supports both synchronous (within metro storage) replication and asynchronous (out of region storage) replication. Particular PureApplication System pattern solutions make use of either, and at times both, synchronous and asynchronous replication, but there are not distinct pattern solutions that allow you chose synchronous or asynchronous. For such a varied selection, you must design a specific solution using patterns and the toolbox.
- ▶ **Active/Active:** The PureApplication System use case scenarios in this book implement inter-rack Active/Active for WebSphere Application Server using split cells across the two racks. A split cell is basically one cluster where the deployment manager and some of the nodes are on one system, and other nodes on another system. This can be extended to a split cell across three or more racks. This scenario can also be extended to racks in different data centers (with the considerations mentioned previously in Geographic dispersion). However, Always On (and resiliency) is typically achieved by providing isolation and independence. When designing for true Always On, a preferred approach is to use independent WebSphere Application Server clusters and cells in each of the data centers (and implement global routing and state replication).

The following PureApplication System use case scenarios described in 1.5, “Overview of HADR use case scenarios” on page 12 can be mapped to the Always On patterns described in Figure 1-6 on page 11:

- ▶ DB2 HADR across two racks
- ▶ WebSphere Application Server cell across two racks in the primary data center
- ▶ Two identical WebSphere Application Server cells across the primary data center and secondary data center
- ▶ Two WebSphere MQ instances (primary and standby) with Multi-Instance Queue Manager (MIQM) across two racks
- ▶ All WebSphere Application Server and DB2 across two separate racks

1.5 Overview of HADR use case scenarios

In this book, the tools in the toolbox are described along with how to implement those tools in the HADR use case scenarios. The following tables provide a brief look at the HADR use case scenarios used in this book. It also provides an in-depth description of these use case scenarios in Chapter 3, “High availability and disaster recovery scenarios” on page 29.

The following list notes the middleware highlighted and the corresponding table:

- ▶ DB2 HADR (see Table 1-1 on page 13)
- ▶ WebSphere Application Server (see Table 1-2 on page 13)
- ▶ WebSphere MQ (see Table 1-3 on page 14)
- ▶ WebSphere Application Server and DB2 (see Table 1-4 on page 15)

Table 1-1 DB2 HADR use case scenarios

| Scenario | Description | When to use the scenario |
|--|--|---|
| DB2 HADR Active/Standby in single rack in the primary data center. | Single DB2 HADR pattern deployed on single rack | Can handle single DB2 (VM or DB2 instance) failure. However, it cannot handle rack or data center failure. |
| DB2 HADR Active/Standby across two racks in the primary data center. | Single DB2 HADR pattern deployed using multi-rack deployment | Can handle single DB2 (VM or DB2 instance) failure, and single rack failure. However, it cannot handle data center failure. |
| DB2 HADR Active/Standby in primary data center and same Active/Passive DR setup in secondary data center | Uses block storage replication | Can handle DB2 (VM or DB2) instance failure, rack failure, and even data center failure. Based on the failure, if the entire data center went down, DB2 HADR instances can be activated on the secondary data center in a short time because the data was being replicated from primary to the secondary data center. |

Table 1-2 WebSphere Application Server use case scenarios

| Scenario | Description | When to use the scenario |
|---|---|---|
| WebSphere Application Server cell in a single rack in the primary data center. | WebSphere cell in the same rack with transaction log stored in GPFS or database, and shared by multiple nodes | Can handle WebSphere node (VM or WebSphere JVM) failures with other nodes handling the requests or recovering the transactions. However, it cannot handle the entire WebSphere cell, or the rack or data center failures. |
| WebSphere Application Server cell across two racks in the primary data center | Transaction logs stored in GPFS or database. Multirack deployment of single pattern across two racks. | Can handle WebSphere node (VM or WebSphere JVM) failures with other nodes handling the requests or recovering the transactions. Can also handle rack failures. However, it cannot handle data center failure. |
| Two identical WebSphere Application Server cells across the primary data center and secondary data center | WebSphere Application Server Active/Passive. Transaction logs stored in GPFS or database. | Can handle single WebSphere node (VM failure, or WebSphere JVM failure), rack failure, and even data center failure. Based on the failure, if the entire data center went down, WebSphere cell can be activated on the secondary data center in a short time because the data was being replicated from primary to the secondary data center. |

Table 1-3 WebSphere MQ use case scenarios

| Scenario | Description | When to use the scenario |
|---|--|---|
| Two WebSphere MQ instances (primary and standby) with MIQM - same rack | Same pattern - Active/Standby in primary data center | Can handle WebSphere MQ primary QManager failure (VM or WebSphere MQ processes) and failures with WebSphere MQ standby handling the messaging. However, it cannot handle entire rack or data center failures. |
| Two WebSphere MQ instances (primary and standby) with MIQM - across two racks | Active/Standby in primary data center. Split pattern for WebSphere MQ. | Can handle WebSphere MQ primary QManager failure (VM or WebSphere MQ processes) and failures with WebSphere MQ standby handling the messaging. Can also handle a rack failure. However, it cannot handle entire rack or data center failures. |
| Two WebSphere MQ instances (primary and standby) with MIQM - across two racks | Active/Standby across data centers. Split pattern for WebSphere MQ. | Can handle WebSphere MQ primary QManager failure (VM or WebSphere MQ processes) and failures with WebSphere MQ standby handling the messaging. Can also handle rack and data center failures. WebSphere MQ can be activated on the secondary data center in a short time because the QManager data was being replicated from primary to the secondary data center. |

Table 1-4 WebSphere Application Server and DB2 use case scenarios

| Scenario | Description | |
|--|--|---|
| WebSphere Application Server cell and DB2 HADR in a single rack in the primary data center. | Transactions are stored in GPFS or database | Can handle WebSphere node (VM or WebSphere JVM) or DB2 (instance of VM) failure with other VMs handling the requests or recovering the transactions. However, it cannot handle the entire WebSphere cell or both DB2 instances, or rack or data center failures. |
| WebSphere Application Server cell across two racks, split cell and DB2 HADR in two separate racks in the primary data center. | Single pattern using multi-rack deployment. Transactions are stored in database. | Can handle WebSphere node (VM or WebSphere JVM) or DB2 (instance or VM) failures with other VMs handling the requests or recovering the transactions. Can also handle rack failures. However, it cannot handle data center failure. |
| WebSphere Application Server Active/Passive, two identical cells across primary data center. DB2 environment replicated on disaster recovery site. | Transactions are stored in GPFS or database. | Can handle single WebSphere node (VM failure, or WebSphere JVM failure) or DB2 (VM or instance) failure, rack failure, and even data center failure. Based on the failure, if the entire data center went down, WebSphere cell and DB2 HADR can be activated on the secondary data center in a short time because the data was being replicated from primary to the secondary data center. |



High availability and disaster recovery capabilities of PureApplication System V2.0

Chapter 1, “Overview” on page 1 provided basic information about the high availability and disaster recovery capabilities of PureApplication System. This chapter provides a more in-depth description of those capabilities.

The following capabilities are described in this chapter:

- ▶ Storage volumes
- ▶ Block storage overview
- ▶ Block storage replication
- ▶ External storage
- ▶ GPFS Overview
- ▶ GPFS topologies
- ▶ Active/Active GPFS deployment
- ▶ Active/Passive GPFS deployment
- ▶ Shared Service for GPFS
- ▶ GPFS file systems and file sets

2.1 Storage volumes

Volumes have been around since PureApplication System V1.0. Simplistically, storage volumes are extra storage that can be attached to virtual machines. For example, you might want extra storage for DB2 to host large databases.

In PureApplication V1.1, two types of storage volumes were available: VMFS and Raw. Virtual Machine File System (VMFS) is specific to the PureApplication System Intel-based system and compatible with the high performance cluster file system. VMFS is primarily used for deployed virtual machines. Raw has no partitions or formatting. Both of these storage volume types are still available in PureApplication System V2.0.

2.1.1 The new type of storage volumes in PureApplication System V2.0

PureApplication System V2.0 introduced a new type of storage volume called *block storage*. Block storage volumes have a separate lifecycle from the pattern instance. This is important because block storage can be added as part of the pattern or added later to the deployed virtual machines. If an application is deleted, the block storage is persisted. Block storage volumes can also be located off-box and accessed as external storage. For more information, see 2.2, “Block storage overview” on page 18.

2.2 Block storage overview

Block storage is declared with the storage area network (SAN) found in PureApplication System. Data is stored in volumes, which are also referred to as blocks. Block storage allows administrators to create volumes of type *block* that allow better access to the SAN inside PureApplication System or an external SAN accessed through the storage volume controller. Block storage provides access to multiple blocks through the file system.

Block storage is used by file systems and database management systems (DBMS). These systems can use block storage as part of providing higher level constructs such as databases and files to programming models. Block storage is not included in application snapshots.

Block storage can be cloned on a single system or replicated to another system for high availability (HA) and disaster recovery (DR) solutions. For internal storage volumes, the maximum internal block size is 8 TB. For external block stores, size depends on external storage support.

Block storage separates the lifecycle of a storage volume and the application (workload). An example is shown in Figure 2-1. If a workload is deleted, the block storage persists. A new workload can be deployed and the persisted block storage can be attached to the new workload. If more storage is required by the workload, you can increase the size of the storage volume.

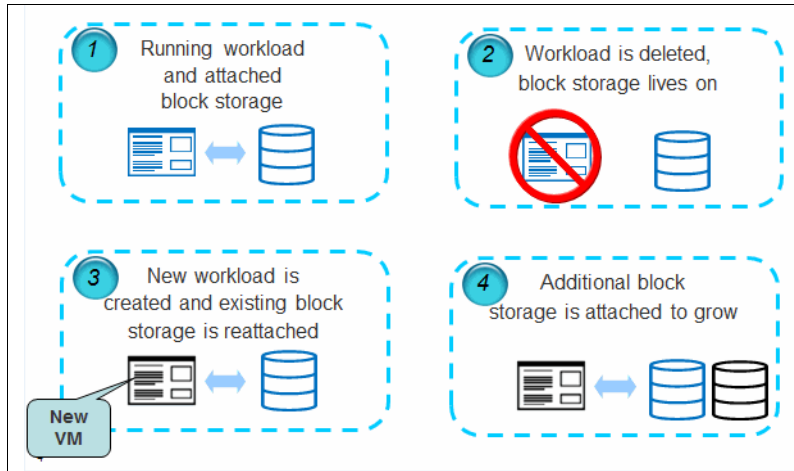


Figure 2-1 Block storage and workload lifecycle separation

Power-based systems and Intel-based systems both support non-shared block storage. Intel-based systems support *block shared storage*. Block shared storage allows one or more virtual machines to share storage volume. Block shared storage provides data sharing between peers. Using shared block storage has several constraints:

- ▶ VMs must have a dedicated compute node
- ▶ VMs must be in the same cloud group
- ▶ VMs must be on the same rack (system)
- ▶ Only GPFS deployments on Intel-based systems use shared block storage

Note: In this publication, the term “rack” refers to a PureApplication System.

2.2.1 Block storage replication

Block storage replication provides the capability of replicating individual block storage volumes to another rack. This is useful for disaster recovery of data where PureApplication System replicates data from a primary rack to another rack in the same (or different) data center.

Storage volumes between the local rack and remote rack are replicated within a consistency group. A *consistency group* is a group of storage volumes of the same type.

Block storage replication needs to be configured in order for replication to occur. A block storage replication profile is required on both local and remote systems. There must be one or more storage volumes of type *block* or *block shared* on both systems. Block storage replication configuration details can be found in 4.2, “Block storage replication configuration” on page 49.

DR for workloads using selective replica across racks

In this disaster recovery scenario, workloads running on the primary rack can use a selective replica on the secondary rack. Several replication pairs can be created between the two racks that are in the block storage replication domain. Several phases outlined in Figure 2-2 illustrate this scenario:

- ▶ Before these phases, block storage replication pairs are created for the storage volumes of the workload.
- ▶ In phase 1, there is a running workload with attached block storage.
- ▶ In phase 2, the storage volumes are replicated.
- ▶ In phase 3, on the remote (DR) rack, deploy a new workload using the replicated storage volume.
- ▶ In phase 4, the storage on the remote rack is attached.

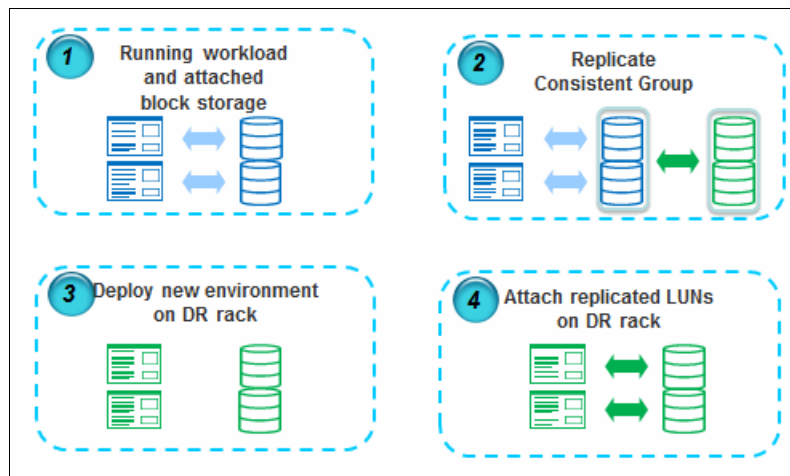


Figure 2-2 DR for workloads using selective replica across racks

A planned failover is an orderly transition of the workload from the sending system to the receiving system. For planned failovers (see 2.3.1, “Planned failover” on page 22) two cases are relevant to this scenario:

- ▶ Stop processing on the local rack and start processing on the remote rack. In this case, there is no data lost. If you break storage volume replication as a group while processing was occurring, you would lose data. This approach requires that the application takes an outage while the application is stopped, replication direction is reversed, and the application is started on the remote rack.
- ▶ Storage volume replication is never stopped. In this case, you do not want to risk data loss. Testing failover is possible on the remote rack using the replication. This approach is suitable for applications that cannot take outages and where data on the remote (DR) rack is acceptable for workload verification.

2.3 Block storage replication

Instead of replicating the entire system configuration for disaster recovery, you can configure block storage replication for disaster recovery at the individual storage volume level. Replication is for active and passive disaster recovery cases.

Replication for each storage volume can be configured between local and remote (recovery) systems in either direction. Replication is configured in volume replication pairs. There is no limit to the number of volume replication pairs. Replication can be configured as synchronous (< 300 kilometers) or asynchronous (< 8000 kilometers). Figure 2-3 illustrates replication pair combinations between a local and a remote rack.

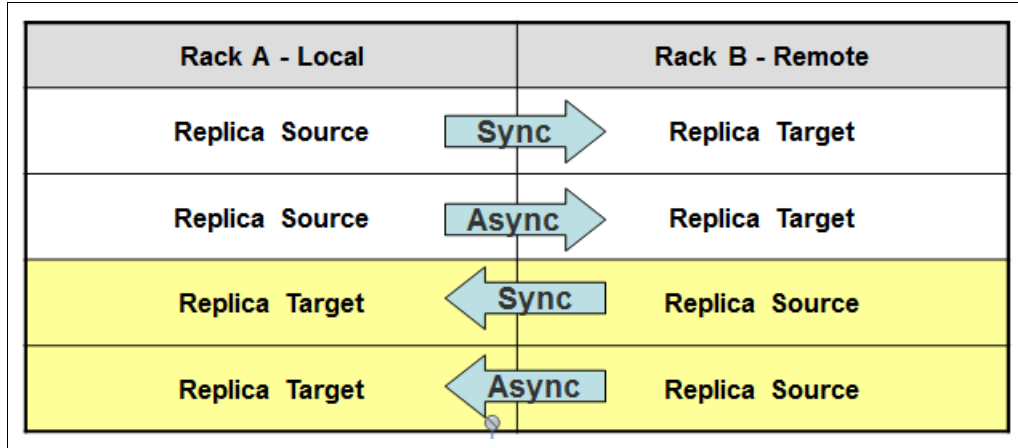


Figure 2-3 Possible replication pairs between local and remote systems

All block storage volumes that have the same source and target, and the same synchronization mode are replicated in an implicit consistency group. Consider the following example:

- ▶ Declare one block storage volume for WebSphere transaction logs and another block storage volume for database data
- ▶ Both are replicated from one rack to another with the same synchronization mode
- ▶ Both are replicated in a consistency group, so the data will be consistent for the workload on the remote DR rack.

During normal production mode, the remote (recovery) rack can be used for other purposes. However, when needed, the remote rack supports both planned and unplanned failovers of the local rack. For more information, see 2.3.1, “Planned failover” on page 22 and 2.3.2, “Unplanned failover” on page 22.

The general procedure for configuring block storage replication is outlined in the following steps. These steps are performed from the System console and are detailed in Chapter 4, “Infrastructure setup” on page 47. The administrator must have permissions for Manage Block Storage (Full) and Manage Security (Full) to configure block storage replication. For a general procedure to configure block storage, use the following steps:

1. Create a storage volume on the primary rack
2. Create a storage volume of same size on the recovery rack.
3. Define block storage replication profiles on the local and remote systems. The profiles establish the relationship between the storage volume pairs for replication.
4. Create the replication pair between the primary and remote storage volumes.
5. Validate the block storage replication profiles on both systems. The validation process verifies:
 - a. Platform compatibility
 - b. Both systems have block storage replication profiles

- c. Peer management (floating IP) properly references the other system
 - d. Identical firmware levels on both systems
 - e. Fibre Channel connectivity between storage controllers on each system
6. Start the block storage replication profiles on both systems.

2.3.1 Planned failover

A planned failover is an orderly transition of the workload from the sending (local) system to the receiving (remote) system. A planned failover is for storage volume recovery.

You can perform a planned failover from a sending (local) system where the local storage volume is replicating to the receiving storage volume on the receiving (remote) system. A block storage replication profile is required on both local and remote systems. There must be one or more storage volumes of type *block* or *block shared* on both systems. Within the PureApplication System, you can run a failover from a block storage replication pair as shown in Figure 2-4.



| Local | Direction | Remote | Replication status | Replication type | Actions |
|--------|-----------|--------|--------------------|------------------|--|
| Test30 | → | Test42 | | Synchronous | <div style="border: 1px solid red; display: inline-block; padding: 2px;">  Failover </div>  Delete |

Figure 2-4 Planned failover run from PureApplication System system console

When you initiate a planned failover, you can determine how the sending and receiving replication roles are handled after the replication completes. The following list notes your options for roles:

- ▶ Replication is enabled in the opposite direction. The sending and receiving replication roles between the storage volumes are “reversed”. This configuration allows you to restore the primary storage after testing the scenario by performing another planned failover from the remote rack.
- ▶ Replication is disabled and replication roles are not swapped. This action allows for a test of an unplanned failover.

2.3.2 Unplanned failover

An unplanned failover is encountered when a real disaster incapacitates a system and makes it inoperable. If block storage replication profiles are enabled, and continuous replication is already in process at the time of the disaster, the only data lost is related to changes that are in process at the time of the disaster. The block replication function allows a system administrator to simulate an unplanned failure.

An unplanned failover is initiated from the receiving (remote) system. You can either have another pattern deployed on the remote system, or deploy an identical pattern on the remote system. In either case, attach the replicated storage to the deployed pattern. The replicated storage contains the application data. This process effectively reestablishes the sending system workload on the receiving system.

Within the PureApplication System, you can run an unplanned failover from a block storage replication profile as shown in Figure 2-5. The replication pair is broken and the block storage on the remote rack becomes active, allowing VMs to attach to that storage.

| Local | Direction | Remote | Replication status | Replication type | Actions |
|--------|-----------|--------|--------------------|------------------|----------|
| Test30 | ← | Test42 | | Synchronous | Failover |

Figure 2-5 Unplanned failover executed from PureApplication System system console.

2.4 External storage

You can configure external storage to one or more PureApplication System racks. You make a connection from your storage volume controller to the SAN connection of the rack. This configuration is analogous to connecting two networks using an Ethernet switch, but instead you are connecting drives instead of computers. An example of this configuration is shown in Figure 2-6.

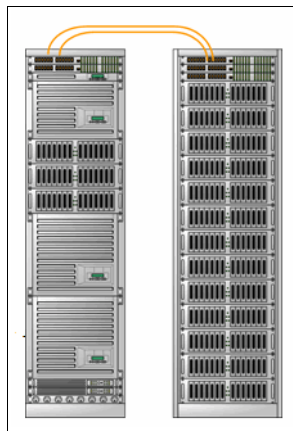


Figure 2-6 Example of external storage and PureApplication System

For a list of supported hardware and procedures to enable the system to use external storage, see the IBM Knowledge Center article “Planning to use external storage” at:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extstorage_plan.dita?lang=en

From an administrative perspective, the external SAN administrator creates storage volumes and shares them with the PureApplication System rack. These external storage volumes are automatically added to PureApplication System. The PureApplication System administrator must enable external storage management from **System Console** → **System** → **Settings**, as shown in Figure 2-7.

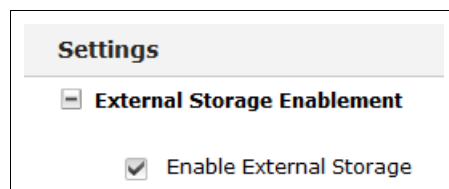


Figure 2-7 External storage enablement

External storage volumes can then be assigned to cloud groups as either block or block shared storage volume types. To attach external storage volumes, the administrator finds the external volume within PureApplication System and assigns the volumes to patterns.

For the patterns, there is no distinction on how it uses storage. The pattern is not aware if the storage is internal or external. External volumes being attached to multiple racks allows high availability across PureApplication System hardware. This is not possible with internal storage.

Note: External storage was not configured for this book.

2.5 GPFS Overview

IBM General Parallel File System (GPFS) is a high performance shared-disk file management system that provides reliable access to multiple nodes in a cluster environment.

GPFS allows the same file to be accessed from multiple different clients. GPFS is built to be redundant so that the file system remains active even if the host nodes become unavailable or inoperable.

IBM PureApplication System provides the IBM Pattern for GPFS, which the system administrator can deploy as a GPFS cluster. The GPFS cluster provides high-performance, highly available storage to the middleware and applications that are deployed in the IBM PureApplication System environment.

The system administrator is responsible for the following actions:

- ▶ Deploying the GPFS Server pattern
- ▶ Attaching storage
- ▶ Defining file systems
- ▶ Deploying the GPFS shared service

One result of this configuration is the GPFS cluster as shown in Figure 2-8. This example configuration is used throughout this book.

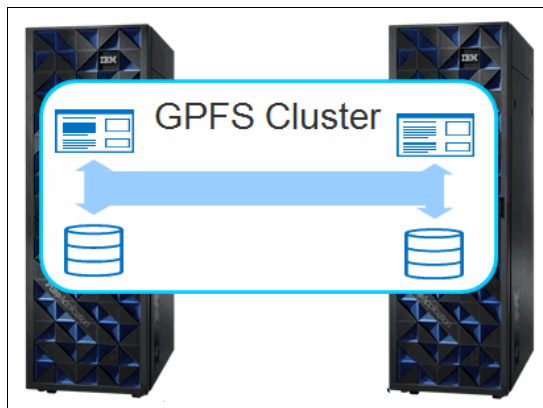


Figure 2-8 GPFS Cluster

2.6 GPFS topologies

Using the IBM Pattern for GPFS, you can deploy a number of different file server topologies. All deployments provide a GPFS Manager virtual machine that directs and controls the underlying GPFS cluster. From the GPFS Manager virtual machine and its application management capabilities, administrators can define file systems, add storage, add file system mirrors, manage security credentials, apply recovery actions in the case of a failure, and attach a passive replica.

The GPFS Server is the first component deployed when planning to use a GPFS shared file system topology. This component is deployed as a virtual application by using the GPFS Pattern Type. When you deploy a GPFS Server component, configure the GPFS Server nodes and attach the list of disks used by the GPFS shared file system.

This virtual application pattern supports the following configurations:

- ▶ Primary
- ▶ Mirror
- ▶ Tiebreaker
- ▶ Passive

For this book, two GPFS server topologies were configured:

- ▶ Active/Active GPFS deployment
- ▶ Active/Passive GPFS deployment

2.7 Active/Active GPFS deployment

The *Active/Active GPFS deployment* is also referred to as active-active mirroring and is considered a high availability scenario. This topology includes a *GPFS Primary* server along with attached *GPFS Mirror* and *GPFS Tiebreaker* servers. The tiebreaker server takes care of quorum and decides which server is primary. The tiebreaker also takes over the primary role of serving clients if either the primary or mirror server becomes unstable. The number of GPFS mirror nodes must match the number of primary nodes.

The size of the mirror and primary storage volumes must match as well. It is interesting to note that the block storage volumes are separate and not aware of each other. GPFS is responsible for the synchronization of data in the two volumes.

Active/Active mirroring is generally used for disaster recovery scenarios over shorter distances (< 300 kilometers). Figure 2-9 shows the three GPFS server configurations deployed to build out this high availability scenario.

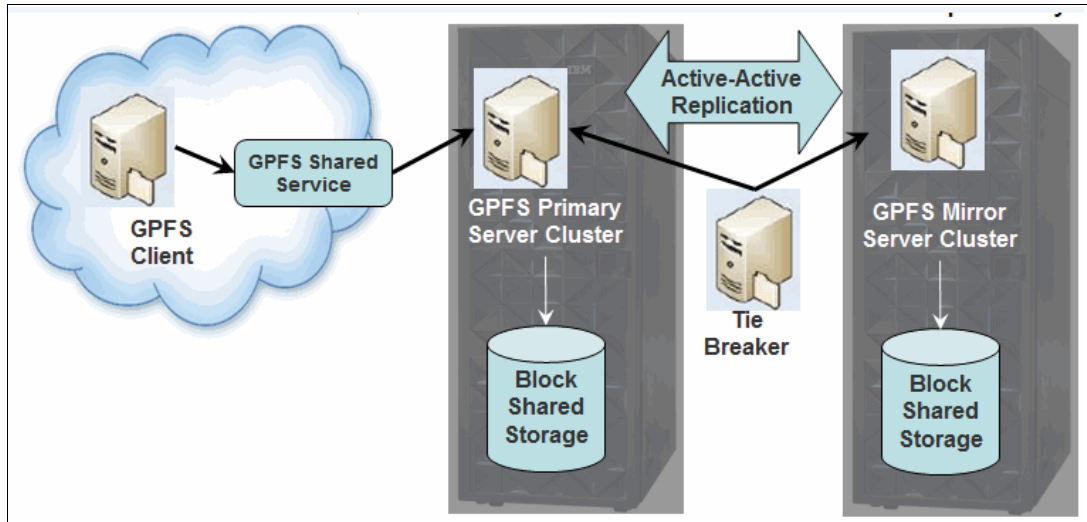


Figure 2-9 Active/Active GPFS deployment

To configure an Active/Active GPFS configuration, use the following steps as a general guide:

1. Create volumes for all three configurations
2. Deploy a mirror server configuration and attach volumes
3. Deploy a tiebreaker server configuration and attach volumes
4. Deploy a primary server configuration and attach volumes
5. Attach mirror and tiebreaker server configurations to the primary configuration

When a failover situation occurs, you can attempt to recover in several ways:

- ▶ If either the Primary or Mirror server configuration fails, you might be able to recover the GPFS cluster.
- ▶ If the Primary or Mirror server configuration fail and the Tiebreaker configuration fails, the GPFS cluster ceases to function.
- ▶ If you lose the Primary configuration, or the Mirror and Tiebreaker configurations, the failure is recoverable.

2.8 Active/Passive GPFS deployment

An *Active/Passive GPFS deployment* includes a GPFS Primary server configuration and a GPFS Passive server configuration. For replication, this configuration uses block storage replication (see 2.3, “Block storage replication” on page 20), which must be set up manually. This configuration, which is shown in Figure 2-10, is generally used over larger distances (< 8000 kilometers).

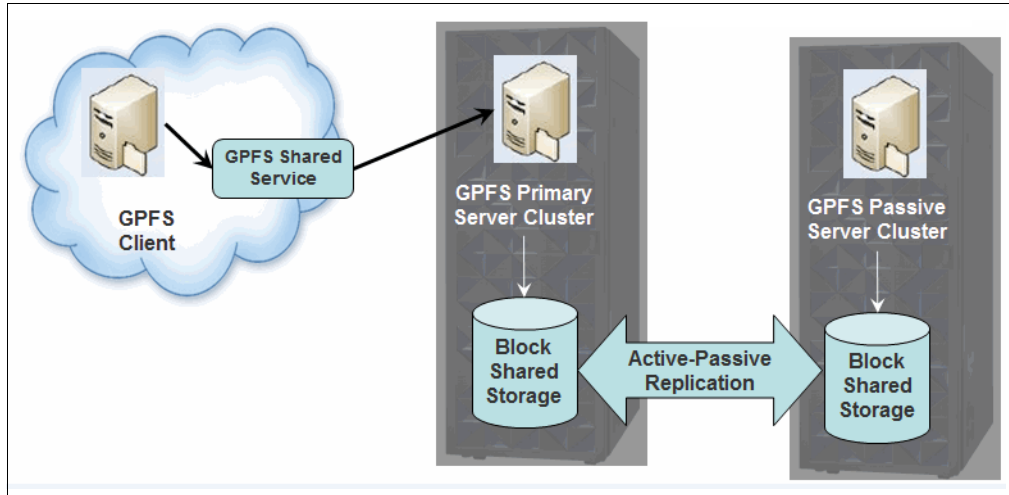


Figure 2-10 Active/Passive GPFS deployment

In the approach shown in Figure 2-10, the two GPFS server configurations are not aware of each other. However, the volumes are aware of each other (synchronized) through block storage replication approach (see 2.3, “Block storage replication” on page 20).

When a planned failover situation is needed, there are several options available (dependent on the state of the primary and remote racks). The options are noted in the following list:

- ▶ If the primary rack is active, perform a failover operation on the replicated storage volumes.
- ▶ If the primary rack is inoperable, run a passive takeover operation to cause the passive server instance to become the new primary configuration.

2.9 Shared Service for GPFS

This shared service is deployed so that GPFS clients in the same cloud group are able to communicate with a deployed GPFS Server. To communicate with a GPFS Server, GPFS clients need to know the client key, which contains the IP address and key retrieved from the Primary GPFS Manager server. The shared service for GPFS simplifies the GPFS client deployments by providing information about available GPFS Servers.

As a prerequisite for this shared service, you must deploy a GPFS Server configuration by using IBM Pattern for GPFS. For more detailed information about GPFS configurations, see 4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53 or 4.4, “Configuring an Active/Passive GPFS deployment” on page 61.

2.10 GPFS file systems and file sets

The GPFS Servers that you create can host multiple *file systems*. File systems are attached to one or more block storage volumes. On the GPFS Server, file systems are mounted as shown in the following commands:

```
/gpfs/fileSystemName1  
/gpfs/fileSystemName2
```

Within each file system, you define file sets that are treated as subdirectories within the file system. File sets are created in the client VM, either through a GPFS client policy or post deployment. Even though file set creation operation is run on the client side, the file set directory is created on the server VMs. A directory is created for each file system and file set in the client VM, as shown in the following commands:

```
/gpfs/fileSystemName1/fileSet1  
/gpfs/fileSystemName1/fileSet2  
/gpfs/fileSystemName2/fileSet1  
/gpfs/fileSystemName2/fileSet2
```

These shared file directories are linked to local file directories. This convention makes it easier for workloads to reference those directories. For example, a shared file directory might look like the following example:

```
/WASTranlogs links to /gpfs/fileSystemName1/fileSet1
```

A maximum of 14 storage volumes can be attached to all file systems for a GPFS Server. File systems must be created before a client can use them.

2.11 Load balancing

Part of any high availability and disaster recovery strategy is the implementation of a load balancer. Load balancers come in the form of hardware and software solutions that allow organizations to distribute inbound traffic across multiple back-end solutions, such as PureApplication System.

In the case where PureApplication System running in the primary data center becomes unavailable, a load balancer can switch inbound traffic to the PureApplication System running in the secondary data center. The load balancer normally has a degree of built-in health monitoring to determine whether the primary data center is not available before routing traffic to the secondary data center.



High availability and disaster recovery scenarios

Though there are many high availability disaster recovery (HADR) scenarios possible, the focus of this book is on some of the most common scenarios across some key IBM middleware products. This book's goal is for you to be able to apply these scenarios to your specific requirements and use cases.

When possible, the new release of the virtual system patterns has been highlighted for some key IBM middleware available with PureApplication System V2. The software components used are for the following releases of the product:

- ▶ WebSphere Application Server V8.5.5.2
- ▶ DB2 V10.5.0.2

In addition to the middleware previously listed, WebSphere MQ V7.5 (available as a Hypervisor Image from PureApplication System V1) is included. As of the release of this book, the new release of WebSphere MQ as a V2 software component is not available.

The focus of this book is about the topology and not the automation of the configuration of the middleware. In production environments, automation by scripts (Urban Code Deploy or other mechanisms) is highly desirable and advised to achieve maximum efficiency from PureApplication System. Every effort is made to identify areas where you can further automate middleware configuration.

The goal is to cover as many scenarios as possible, but due to limited time, only a subset of scenarios is covered. Continued coverage of other scenarios and new solutions will be offered in future publications such as revisions to this publication, DeveloperWorks articles, and so on.

This chapter includes the following sections:

- ▶ Overview for the scenarios
- ▶ HADR scenarios for WebSphere Application Server
- ▶ HADR scenarios for DB2
- ▶ HADR scenarios for WebSphere Application Server and DB2
- ▶ HADR scenarios for WebSphere MQ

3.1 Overview for the scenarios

The following information sections are provided to give clarity to the scenarios covered in this book:

- ▶ Nomenclature
- ▶ Patterns
- ▶ Rack topology
- ▶ PureApplication Platform for testing scenarios
- ▶ Scenario basics

3.1.1 Nomenclature

The names that are used in the HADR scenarios are noted in the following list:

- ▶ PDC: Primary data center
- ▶ SDC: Secondary data center (also Disaster Recovery data center in this book)
- ▶ Racks:
 - PDC-1: Rack 1 in PDC
 - PDC-2: Rack 2 in PDC
 - SDC-1: Rack 1 in SDC
 - SDC-2: Rack 2 in SDC

3.1.2 Patterns

The scenarios contain one or more virtual system patterns. The topology examples (shown in the figures within this chapter) display the boundaries of the virtual system patterns that are used to create the scenarios.

3.1.3 Rack topology

The rack topology used for these scenarios is noted in the following list:

- ▶ For most active scenarios within the data center: One rack is assigned as PDC-1 and another rack is assigned as PDC-2 (in the same data center or a data center in close proximity (< 300 km)).
- ▶ For active/passive scenarios across data centers: One rack is assigned as PDC-1 in the first data center, and another rack (mimicking the DR rack) in the second data center is assigned as SDC-1.

For scenarios with two data centers, the centers can be in close proximity (< 300 km) or separated over a larger distance (< 8000 km). For close proximity data centers, where block storage replication is used, synchronous replication mode is used. For data centers that are separated over larger distances, asynchronous mode replication must be used. In this chapter's example, *sync* mode is used, but the same scenarios can be set with *async* mode.

If additional PureApplication System racks are available, a golden HADR deployment provides a more robust HADR solution. In the golden HADR deployment, there are two racks in the PDC and two racks in the SDC (or if two racks are not available, then one rack in the SDC).

3.1.4 PureApplication Platform for testing scenarios

Each of the scenarios were tested on a specific PureApplication platform (Intel or Power). However, they are expected to run on either platform without modification. In the detailed section for each scenario, the platforms that were tested are listed. Some of them were tested using the Intel platform, whereas others were tested using the Power platform.

3.1.5 Scenario basics

Based on the selected middleware, the following list notes the highlighted scenarios:

- ▶ WebSphere Application Server
- ▶ DB2
- ▶ WebSphere Application Server and DB2
- ▶ WebSphere MQ

In each of the scenario descriptions, a list is provided of new HADR functions that are used (GPFS, Block Storage). To test the scenarios, the following listed test cases are used:

- ▶ For WebSphere and DB2 scenarios, a small test case was created, called *Bank Transaction* that interacts with two databases. The databases (checking and savings) are tested to allow deposit, withdrawal, and transfer from those accounts. Testing uses a 2-phase commit transaction for moving data between the two databases. The test case details are provided in Appendix A, “Sample Application” on page 253, and is available for download (as-is) as detailed in Appendix C, “Additional material” on page 287.
- ▶ For WebSphere MQ, the samples `amqspout` and `amqsget` that are included with WebSphere MQ are used. These samples allow put and retrieve of messages from the queue.

Each of the following sections provides more details of the architecture and topology.

3.2 HADR scenarios for WebSphere Application Server

Table 3-1 lists all the WebSphere Application Server scenarios described in this chapter and gives additional specifics for how these scenarios function.

Table 3-1 WebSphere Application Server Scenarios

| Scenario name | Description | New HADR features used in the scenario | Platform tested | Rack: PDC-1 | Rack: PDC-2 | Rack: SDC-1 | Rack: SDC-2 |
|---------------|---|---|-----------------|-------------|-------------|-------------|-------------|
| WAS_1 | WebSphere cell in the same rack with transaction log stored in GPFS | GPFS Primary configuration | Power | x | | | |
| WAS_2 | WebSphere cell across multiple racks in the same data center | Multi-rack deployment GPFS Primary-Mirror configuration across two racks. | Power | x | x | | |

| Scenario name | Description | New HADR features used in the scenario | Platform tested | Rack: PDC-1 | Rack: PDC-2 | Rack: SDC-1 | Rack: SDC-2 |
|---------------|--|---|-----------------|-------------|-------------|-------------|-------------|
| WAS_3 | WebSphere active-passive cells with active cell in PDC and identical passive cell in SDC Recovery of the transactions on DR site (SDC) requires that the WebSphere nodes in the DR site have the same host names. | Multi-rack deployments. GPFS Passive configuration for WebSphere transactions, across two racks. | Power | x | x | x | x |

The middleware used in these scenarios is WebSphere Application Server V8.5.5. The topology diagram (of each of the scenarios that are listed in Table 3-1 on page 31) are outlined in the following sections that discuss each scenario by name:

- ▶ 3.2.1, “Scenario WAS_1: WebSphere cell in the same rack (PDC-1) with transactions in GPFS” on page 32
- ▶ 3.2.2, “Scenario WAS_2: WebSphere cell across two racks in same data center” on page 33
- ▶ 3.2.3, “Scenario WAS_3: WebSphere active-passive cells - identical setup in PDC and SDC, with WebSphere transactions stored in GPFS” on page 34

3.2.1 Scenario WAS_1: WebSphere cell in the same rack (PDC-1) with transactions in GPFS

Figure 3-1 shows the topology diagram for this scenario.

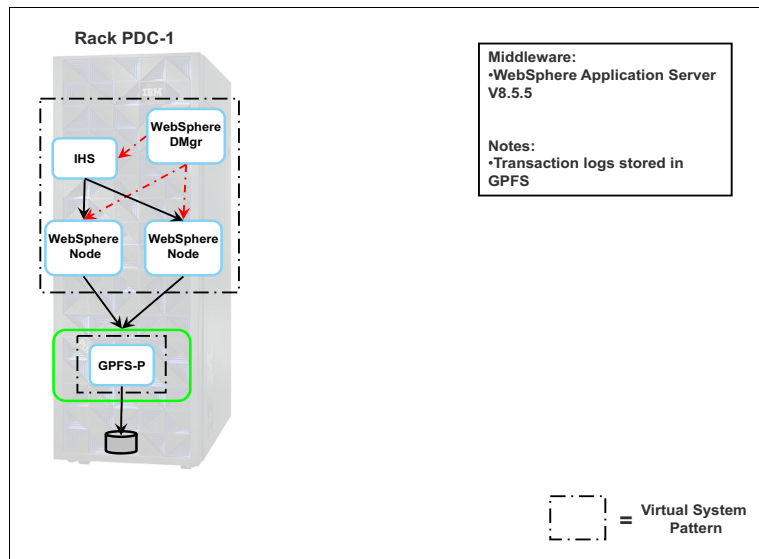


Figure 3-1 Scenario WAS_1

The following list notes the benefits and considerations of scenario WAS_1:

- ▶ Benefits:
 - HA available within the single rack
 - Most of this pattern is built from pre-built setup features, except the configuration of the application server
- ▶ Considerations:
 - Does not tolerate rack failure
 - Does not tolerate GPFS Primary node failure (can have GPFS Mirror configuration on the same rack to tolerate GPFS Primary node failure)

Additional information for scenario WAS_1 is noted in the following list:

- ▶ Test case used: Bank Transaction sample
- ▶ Possible automation: Automate the WebSphere configuration for transactions directory to point to GPFS directory

3.2.2 Scenario WAS_2: WebSphere cell across two racks in same data center

Figure 3-2 shows the topology diagram for this scenario.

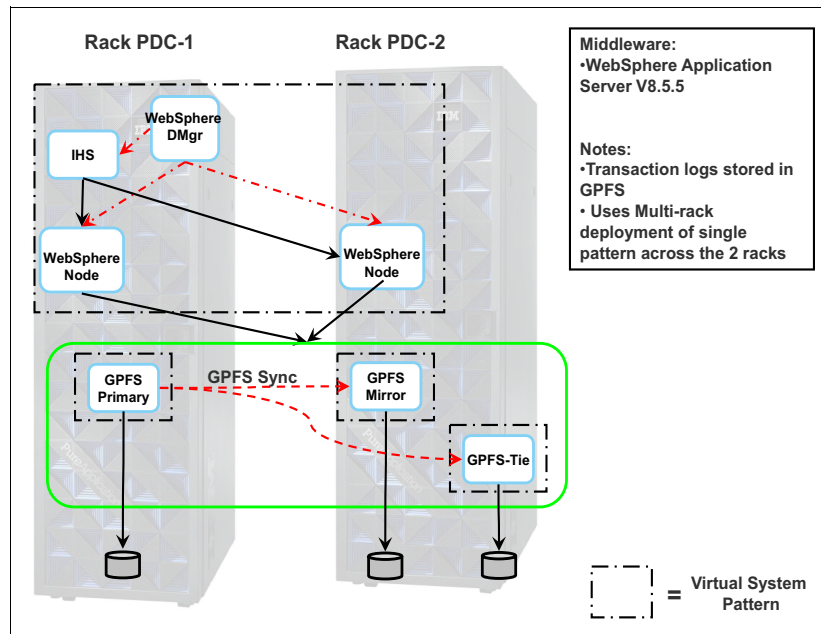


Figure 3-2 Scenario WAS_2

The following list notes the benefits and considerations of scenario WAS_2:

- ▶ Benefits:
 - HA available across the racks, and hence tolerates single rack failure
 - Tolerates GPFS node failure on a single rack
 - Most of this pattern is built from pre-built setup features, except the configuration of the application server

- ▶ Considerations:
 - Does not tolerate entire data center failure
 - Tie breaker needs to be on a separate environment to avoid single rack failure

Additional information for scenario WAS_2 is noted in the following list:

- ▶ Test case used: Bank Transaction sample.
- ▶ Possible automation: Automate the WebSphere configuration for transactions directory to point to GPFS directory.

3.2.3 Scenario WAS_3: WebSphere active-passive cells - identical setup in PDC and SDC, with WebSphere transactions stored in GPFS

Figure 3-3 shows the topology diagram for this scenario.

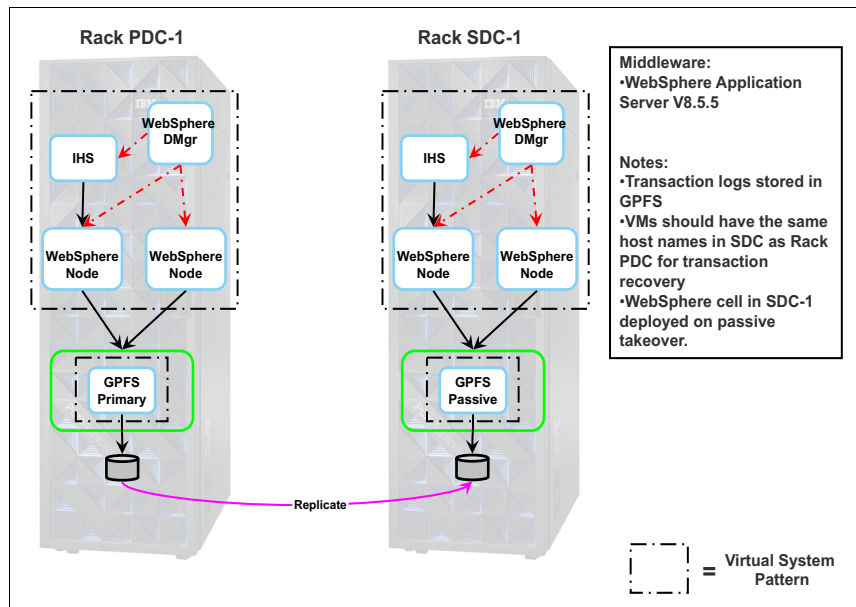


Figure 3-3 Scenario WAS_3

Recovery of transactions by WebSphere nodes in SDC requires that the WebSphere VMs have the same host name as the VMs in the PDC. The network team needs to ensure that the DNS in SDC is set up to provide same host names. The VMs in the SDC are only started during the activation of the DR, so having the same host names does not create any conflicts.

The following list notes the benefits and considerations of scenario WAS_3:

- ▶ Benefits:
 - HA available within the rack and DR across data centers. Tolerates data center failure
 - Most of this pattern is built from pre-built setup features, except the configuration of the application server
- ▶ Considerations:
 - Need to activate DR when there is a rack failure in PDC

Note: If you have the luxury of having a golden HADR setup, you can combine scenarios WAS_2 and WAS_3. The only change is that you have GPFS Passive configuration instead of a combined GPFS Mirror and Passive deployments.

Additional information for scenario WAS_3 is noted in the following list:

- ▶ Test case used: Bank Transaction sample.
- ▶ Possible automation: Automate the WebSphere configuration for transactions directory to point to GPFS directory.

3.3 HADR scenarios for DB2

Table 3-2 lists the HADR scenarios for DB2 V10.5.2 that are the focus for this book and gives more detailed specifics for the scenarios.

Table 3-2 DB2 scenarios

| Scenario name | Description | New HADR features used in the scenario | Platform tested | Rack: PDC-1 | Rack: PDC-2 | Rack: SDC-1 | Rack: SDC-2 |
|---------------|--|---|-----------------|-------------|-------------|-------------|-------------|
| DB2_1 | DB2 HADR from the same pattern and deployed in a single rack | N/A | Intel | x | | | |
| DB2_2 | DB2 HADR from the same pattern and deployed across two racks using multi-rack deployment | Multi-rack deployment | Intel | x | x | | |
| DB2_3 | Two identical DB2 HADR deployments across two different data centers, PDC and SDC, in which the SDC deployment provides the DR for any data center failure in PDC. | Multi-rack deployment and block storage replication | Intel | x | | x | |

The following list notes additional details about the scenarios that are listed in Table 3-2:

- ▶ The DB2 HADR solution consists of Active-Standby DB2 instances.
- ▶ The data replication across the DB2 instances is managed by DB2, using different data synchronization modes (such as Synchronous, Near-synchronous, Asynchronous, and Super-asynchronous). The DB2 HADR pattern (a pre-built feature ready for use) supports Asynchronous and Near-synchronous modes.
- ▶ DB2 clients can use the automatic client routing feature. This feature detects any failure in the active DB2 instance and sends requests to the standby DB2 instance.

A more detailed explanation of the topology for each of the scenarios in Table 3-2 on page 35 are described in the following sections:

- ▶ 3.3.1, “Scenario DB2_1: DB2 HADR from the same pattern and deployed on a single rack (PDC-1)” on page 36
- ▶ 3.3.2, “Scenario DB2_2: DB2 HADR from the same pattern and deployed the parts on two racks (PDC-1 and PDC-2)” on page 37
- ▶ 3.3.3, “Scenario DB2_3: Identical DB2 HADR deployments across primary (PDC) and secondary DR (SDC) data centers” on page 38

3.3.1 Scenario DB2_1: DB2 HADR from the same pattern and deployed on a single rack (PDC-1)

Figure 3-4 shows the topology diagram for this scenario.

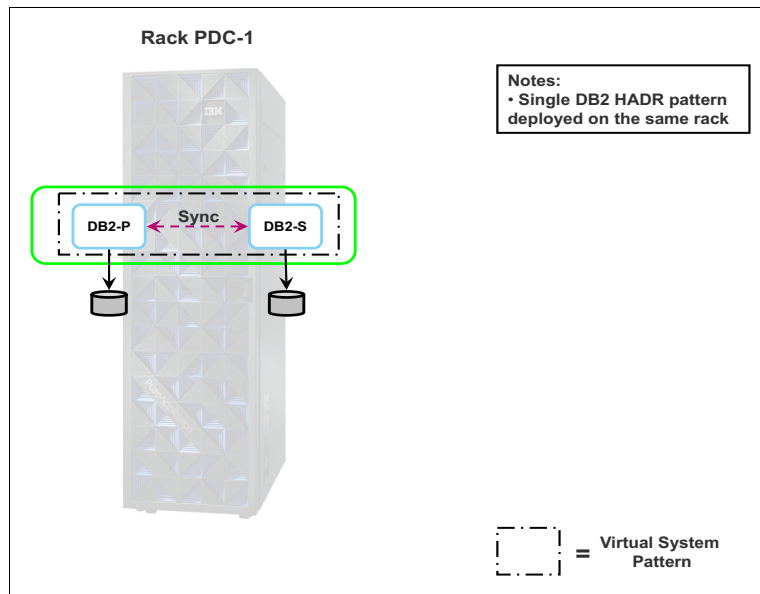


Figure 3-4 Scenario DB2_1

The following list notes the benefits and considerations of scenario DB2_1:

- ▶ Benefits:
 - Active-Standby available within the single rack
 - Most of this pattern is built from pre-built setup features
- ▶ Considerations:
 - Does not tolerate rack failure

Additional information for scenario DB2_1 is noted in the following list:

- ▶ Test case used: Bank Transaction sample with WebSphere Application Server
- ▶ Possible automation: Not applicable

3.3.2 Scenario DB2_2: DB2 HADR from the same pattern and deployed the parts on two racks (PDC-1 and PDC-2)

Figure 3-5 shows the topology diagram for this scenario.

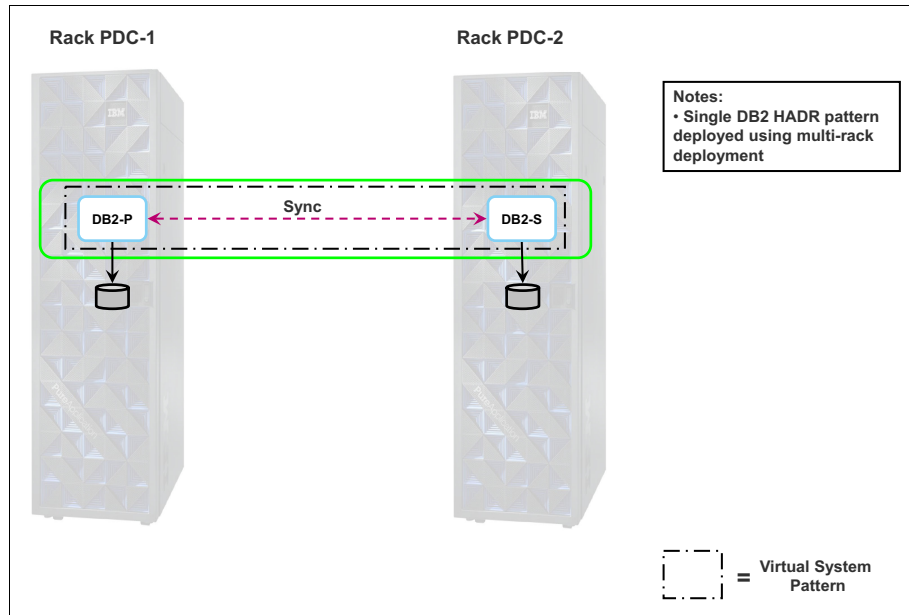


Figure 3-5 Scenario DB2_2

The following list notes the benefits and considerations of scenario DB2_2:

- ▶ Benefits:
 - Active-Standby available across the racks, so this configuration tolerates rack failure
 - Most of this pattern is built from pre-built setup features
- ▶ Considerations:
 - Requires multi-rack deployment domain set up
 - Does not tolerate data center failure where both the racks are affected

Additional information for scenario DB2_2 is noted in the following list:

- ▶ Test case used: Bank transaction sample with WebSphere Application Server
- ▶ Possible automation: Not applicable

3.3.3 Scenario DB2_3: Identical DB2 HADR deployments across primary (PDC) and secondary DR (SDC) data centers

Figure 3-6 shows the topology diagram for this scenario.

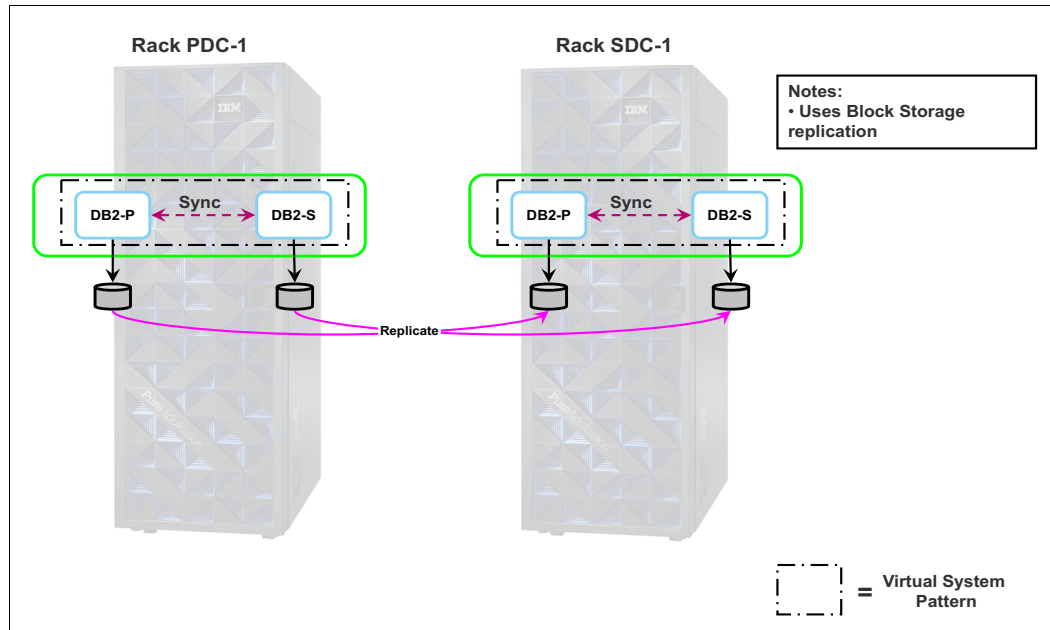


Figure 3-6 Scenario DB2_3

The following list notes the benefits and considerations of scenario DB2_3:

- ▶ Benefits:
 - Active-Standby available within the single rack
 - Tolerates data center failure
- ▶ Considerations:
 - Requires Block storage replication if internal block storage is used

Additional information for scenario DB2_3 is noted in the following list:

- ▶ Test case used: Bank transaction sample with WebSphere Application Server
- ▶ Possible automation: Not applicable

3.4 HADR scenarios for WebSphere Application Server and DB2

Table 3-3 lists all the WebSphere Application Server and DB2 scenarios that are highlighted in this publication. The middleware used in these scenarios is WebSphere Application Server V8.5.5 and DB2 V10.5.

Table 3-3 WebSphere Application Server and DB2 scenarios

| Scenario name | Description | New HADR features used in the scenario | Platform tested | Rack: PDC-1 | Rack: PDC-2 | Rack: SDC-1 | Rack: SDC-2 |
|---------------|---|--|-----------------|-------------|-------------|-------------|-------------|
| WDB_1 | WebSphere Application Server cluster and DB2 HADR deployed on a single rack, with transactions stored in the database | | Power | x | | | |
| WDB_2 | WebSphere Application Server cluster split across two racks in the same data center with DB2 HADR also split across the racks. WebSphere transactions are stored in the database. | | Power | x | x | | |
| WDB_3 | Identical WebSphere Application Server cell and DB2 HADR replicated across DR site. | Block storage replication | Power | x | | x | |

The topology diagram of each of the mentioned scenarios in Table 3-3 are covered in more detail in the following sections:

- ▶ 3.4.1, “Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database” on page 40
- ▶ 3.4.2, “Scenario WDB_2: WebSphere Application Server cluster split across two racks in the same data center with DB2 HADR also split across the racks, with WebSphere transactions stored in database” on page 41
- ▶ 3.4.3, “Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in DB” on page 42

3.4.1 Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database

Figure 3-7 shows the topology diagram for this scenario.

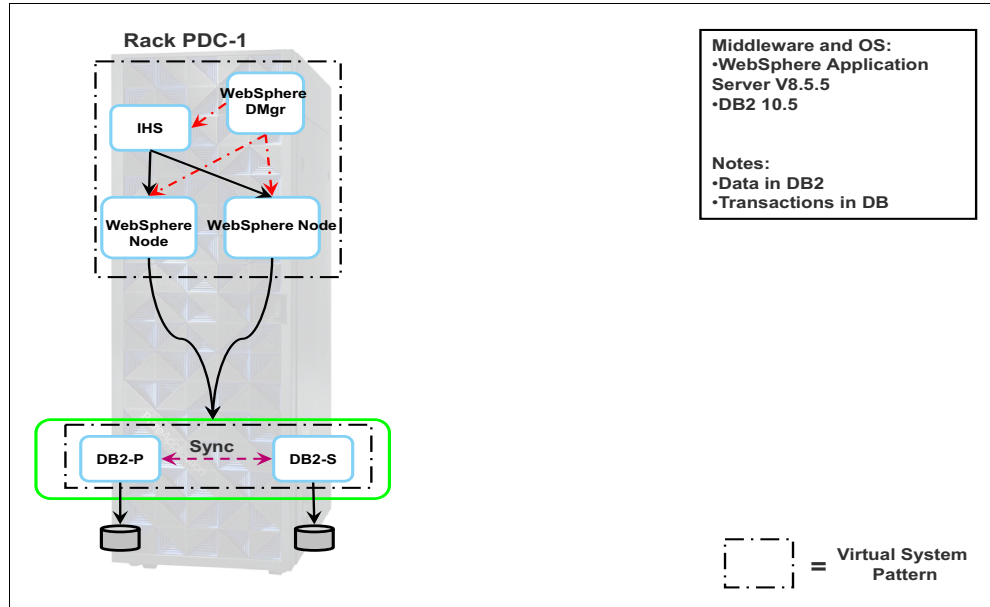


Figure 3-7 Scenario WDB_1

The following list notes the benefits and considerations of scenario WDB_1:

- ▶ **Benefits:**
 - Active WebSphere nodes are in the same cell.
 - DB2 HADR active and standby in the same rack.
 - Most of this pattern is built from pre-built setup features.
 - Stores WebSphere transactions in database. Reduces setup process because a shared files system is not needed for WebSphere transactions
- ▶ **Considerations:**
 - Does not tolerate rack failure

Additional information for scenario WDB-1b is noted in the following list:

- ▶ Test case used: Bank Transaction failure
- ▶ Possible automation: Not applicable

3.4.2 Scenario WDB_2: WebSphere Application Server cluster split across two racks in the same data center with DB2 HADR also split across the racks, with WebSphere transactions stored in database

Figure 3-8 shows the topology diagram for this scenario.

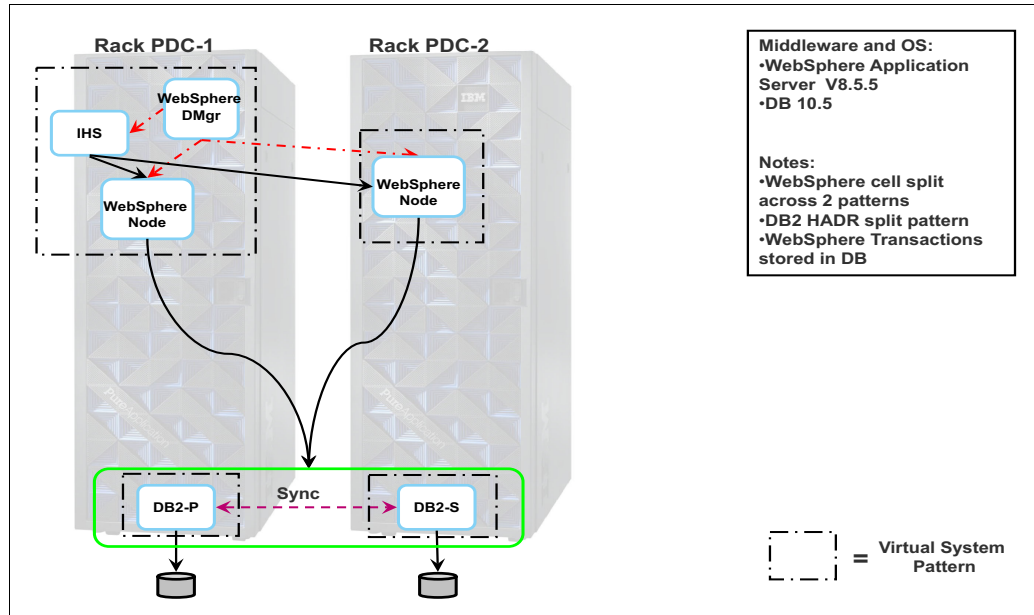


Figure 3-8 Scenario WDB_2

The following list notes the benefits and considerations of scenario WDB_2:

- ▶ **Benefits:**
 - Active WebSphere nodes split across racks in the same data center.
 - DB2 HADR active and standby across multiple racks.
 - Most of this pattern is built from pre-built setup features.
 - Stores WebSphere transactions in database. Reduces setup process as shared file system is not needed for WebSphere transactions.
 - Tolerates single rack failure
- ▶ **Considerations:**
 - Does not tolerate data center failure.

Additional information for scenario WDB_2 is noted in the following list:

- ▶ Test case used: Bank Transaction sample
- ▶ Possible automation: Not applicable

3.4.3 Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in DB

Figure 3-9 shows the topology diagram for this scenario.

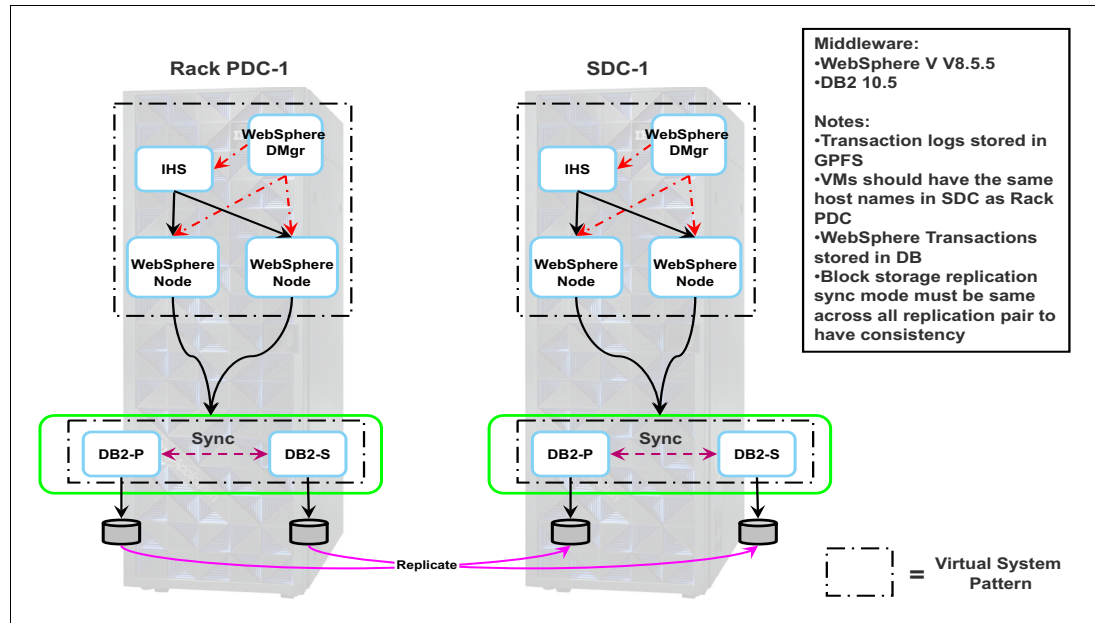


Figure 3-9 Scenario WDB_3

The WebSphere cell in the DR site can be provisioned as noted in the following list:

- ▶ After passive takeover
- ▶ Pre-provisioned with application servers and DB2 being stopped, to be started after the passive failover of the block storage on the DR rack

The following list notes the benefits and considerations of scenario WDB_3:

- ▶ Benefits:
 - Identical WebSphere cell and DB2 setup across two sites, providing a recovery from a complete data center failure.
 - Most of this pattern is built from pre-built setup features
- ▶ Considerations:
 - WebSphere cell transaction recovery requires identical host names for WebSphere cell on the DR site
 - Some manual steps are required for passive takeover and starting of DB2 and WebSphere on the DR site (the time required can be 2 hours or less, based on how much has been pre-provisioned)

Additional information for scenario WDB_3 is noted in the following list:

- ▶ Test case used: Bank Transaction sample
- ▶ Possible automation: Not applicable

3.5 HADR scenarios for WebSphere MQ

Table 3-4 lists the HADR scenarios for WebSphere MQ V7.5 Hypervisor image using Multi-Instance Queue Manager (MIQM) for this publication.

Table 3-4 WebSphere MQ scenarios

| Scenario name | Description | New HADR features used in the scenario | Platform tested | Rack: PDC-1 | Rack: PDC-2 | Rack: SDC-1 | Rack: SDC-2 |
|---------------|--|---|-----------------|-------------|-------------|-------------|-------------|
| WMQ_1 | WebSphere MQ primary and standby parts in the same pattern deployed in a single rack | GPFS-Primary configuration sharing QManager files (data, logs, and error files) | Intel | x | | | |
| WMQ_2 | WebSphere MQ primary and standby parts in the different patterns deployed on two different racks within the same data center (or can even be across data centers with distance less than 300 km) | GPFS-Mirror configuration for sharing QManager files | Intel | x | x | | |
| WMQ_3 | WebSphere MQ primary and standby parts in the different patterns deployed on two different racks across data centers over large geographic distances | GPFS-Passive configuration for sharing QManager files | Intel | x | | x | |

The following is additional information about the scenarios noted in Table 3-4:

- ▶ WebSphere MQ MIQM requires a shared file system to share the Queue Manager (QMGr) files (data, logs, and errors) across the different MIQM Queue Managers. When the primary QMGr fails, the standby automatically becomes active and because it is using the sharing of the file systems, it has access to the Queue Manager data.
- ▶ There are other HADR solutions, such as using the WebSphere MQ cluster solution. Though this chapter focuses on MIQM, some of the base concepts (used within these scenarios for shared file systems) can be used for other HADR solutions.
- ▶ The HADR solution consists of an active-standby WebSphere MQ QManager instance.

The topology diagram of each of the mentioned scenarios in Table 3-4 are covered in more detail in the following sections:

- ▶ 3.5.1, “Scenario WMQ_1: WebSphere MQ primary and standby in the same pattern deployed on a single rack” on page 44
- ▶ 3.5.2, “Scenario WMQ_2: WebSphere MQ primary and standby in different pattern deployed on two different racks within the same data center” on page 45
- ▶ 3.5.3, “Scenario WMQ_3: WebSphere MQ primary and passive in the different patterns deployed on separate racks across the data center” on page 46

3.5.1 Scenario WMQ_1: WebSphere MQ primary and standby in the same pattern deployed on a single rack

Figure 3-10 shows the topology diagram for this scenario.

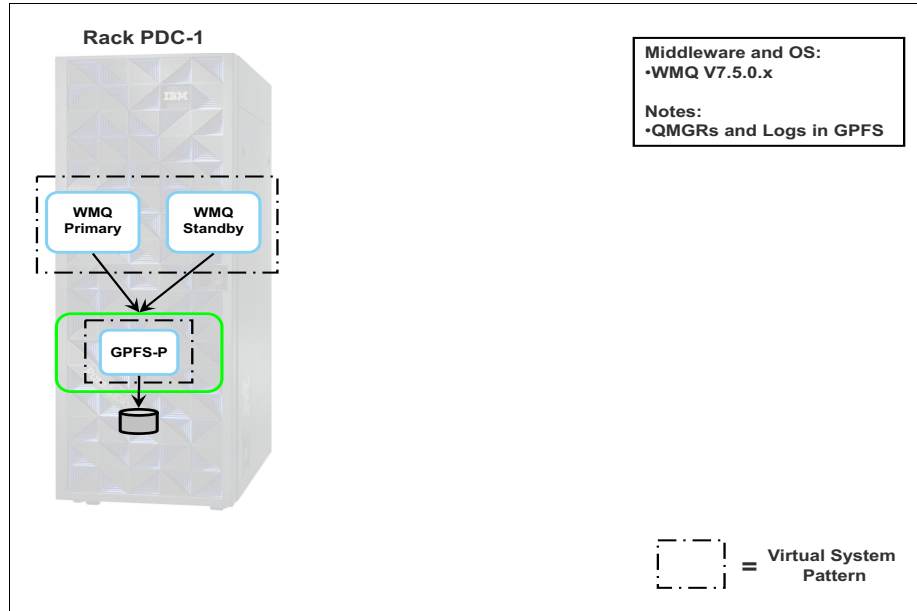


Figure 3-10 Scenario WMQ_1

The following list notes the benefits and considerations of scenario WMQ_1:

- ▶ Benefits:
 - Active-Standby available within the single rack
 - Most of this pattern is built from pre-built setup features
- ▶ Considerations:
 - Does not tolerate rack failure

Additional information for scenario WMQ_1 is noted in the following list:

- ▶ Test case used: Built-in product samples:
 - **amqspu**t: To put messages to a queue associated with a QManager
 - **amqsge**t: To get messages from the queue
- ▶ Possible automation: Not applicable

3.5.2 Scenario WMQ_2: WebSphere MQ primary and standby in different pattern deployed on two different racks within the same data center

This scenario can also be used for racks in different data centers within close geographic locations. It uses GPFS mirror con-figuration across the two racks. Figure 3-11 shows the topology diagram for this scenario.

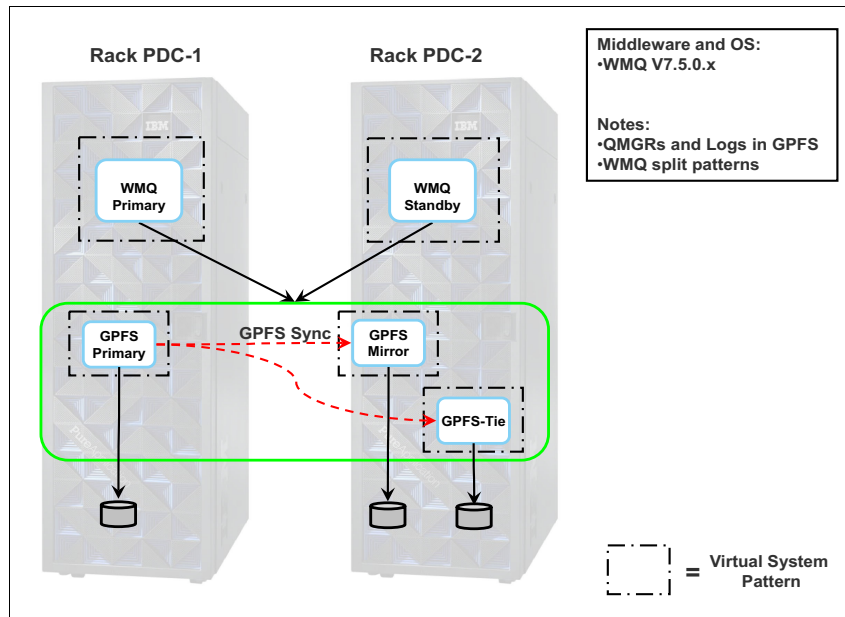


Figure 3-11 Scenario WMQ_2

The following list notes the benefits and considerations of scenario WMQ_2:

- ▶ Benefits:
 - Active-Standby available across racks
 - GPFS mirror configuration used so that a GPFS node failure in one rack is tolerated
- ▶ Considerations:
 - Does not tolerate data center failure

Additional information for scenario WMQ_2 is noted in the following list:

- ▶ Test case used: Built-in product samples:
 - **amqspu**t: To put messages to a queue associated with a QManager
 - **amqsge**t: To get messages from the queue
- ▶ Possible automation: Not applicable

3.5.3 Scenario WMQ_3: WebSphere MQ primary and passive in the different patterns deployed on separate racks across the data center

This scenario is for racks in different data centers across large geographic distances. GPFS passive configuration is used across the two data centers. Figure 3-12 shows the topology diagram for this scenario.

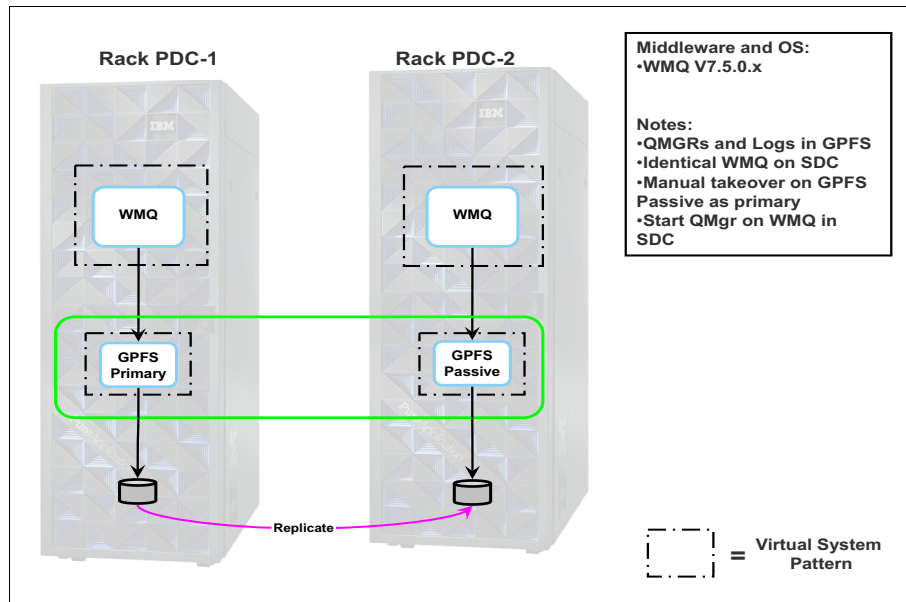


Figure 3-12 Scenario WMQ_3

The following list notes the benefits and considerations of scenario WMQ_3:

- ▶ **Benefits:**
 - Tolerates data center failure.
 - Most of this pattern is built from pre-built setup features.
- ▶ **Considerations:**
 - The second WebSphere MQ instance is a passive one. The GPFS passive instance needs to be activated for the second instance of WebSphere MQ to process the messages.

Additional information for scenario WMQ_3 is noted in the following list:

- ▶ **Test case used: Built-in product samples:**
 - **amqsput:** To put messages to a queue associated with a QManager
 - **amqsget:** To get messages from the queue
- ▶ **Possible automation: Not applicable**



Infrastructure setup

This chapter describes specific details about the infrastructure setup used to configure and test the scenarios in this book. You can find details about those scenarios in Chapter 3, “High availability and disaster recovery scenarios” on page 29.

The following topics are covered in this chapter:

- ▶ Block storage configuration
- ▶ Block storage replication configuration
- ▶ Configuring an Active/Active (Mirrored) GPFS deployment
- ▶ Configuring an Active/Passive GPFS deployment
- ▶ Deploy GPFS Shared Service
- ▶ External storage configuration
- ▶ Network configuration and cloud resources configuration
- ▶ Multisystem environment deployment
- ▶ DNS setup for primary and secondary (cross) rack scenarios
- ▶ Network configuration for WebSphere Application Server and DB2 scenarios

4.1 Block storage configuration

Configure block storage early during the development of your high availability (HA) and disaster recovery (DR) strategy implementation. The block storage configuration that is described here was used by the Active/Active General Parallel File System (GPFS) deployment (4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53) and by the Active/Passive GPFS deployment (4.4, “Configuring an Active/Passive GPFS deployment” on page 61).

For an overview of block storage, see Chapter 2, “High availability and disaster recovery capabilities of PureApplication System V2.0” on page 17.

You can configure multiple systems, but for the block storage configuration in this book, two PureApplication System systems were involved: Primary and secondary. On both systems, the number and size of the volumes is identical. This is a requirement if you are configuring either an active/active GPFS deployment or an active/passive GPFS deployment.

Because the primary and secondary racks were used for both GPFS configurations, several block storage volumes were created for use in these configurations. The example shown in Table 4-1 and Table 4-2 are for an Intel system, so the type is **shared block**. On IBM Power Systems™, the type is **block**.

Table 4-1 Primary rack storage volume specification

| Volume name | Size | Type |
|---------------------|--------|--------------|
| RBHADR_GPFS_Primary | 100 GB | Shared block |

Table 4-2 Secondary storage volume specifications

| Volume name | Size | Type |
|-----------------------------------|--------|--------------|
| RBHADR_GPFS_Mirror | 100 GB | Shared block |
| RBHADR_GPFS_Tie | 20 GB | Shared block |
| RBHADR_GPFS_PassiveConfig_Passive | 100 GB | Shared block |

4.1.1 Block storage configuration

The steps to configure block storage are outlined below. This block storage configuration was used for the GPFS Primary configuration on the primary rack.

1. From the System Console, select **Cloud** → **Storage Volumes**.
2. Click **Create New**.

3. Create the storage volume by entering appropriate values. For this example (Figure 4-1), a 100 GB block shared volume was created in an externally managed cloud group.

Create a new storage volume

* Name:

Description:

Type:

Storage Volume Group:

* In Cloud Group:

Volume configuration: Existing settings Customize settings

* Size:

Figure 4-1 Example of block shared storage volume configuration

4. Create the remaining block storage volumes that are needed for the other GPFS configurations (Tiebreaker, Mirror, and Passive) on the remote systems as specified in Table 4-2 on page 48.

4.2 Block storage replication configuration

Block storage replication is configured between storage volumes on different racks. For this particular configuration, block storage replication is configured between the storage volumes used by the GPFS Primary configuration (primary rack) and GPFS Passive configuration (secondary rack).

Before starting, verify that each rack has the appropriate storage volumes of the same size and type.

The glue between the storage volumes is a **block storage replication profile** on each participating rack, which defines the storage volumes that are replicated as part of the block storage replication process.

4.2.1 Block storage replication: Steps

The following are the steps to configure block storage replication profiles on the primary and secondary racks.

To configure a block storage replication profile on the primary rack, complete these steps:

1. From the System Console, select **System** → **Block Storage Replication**.
2. Click the **New** icon.

3. Complete these required fields:
 - a. **Name:** Provide a descriptive name such as “Redbooks-DR.”
 - b. **Peer management location:** This is the management IP address or fully qualified domain name (FQDN) of the system to which a relationship is established (Passive rack).

Note: Use the FQDN instead of an IP address. You can locate the management IP address for the Passive system by clicking **System** → **Network Configuration**. Expand the Management IP section. Use the floating IP address.

4. **Trust User ID:** This user ID establishes the trust relationship between the host and remote system. The user must have full permissions to manage security.
 - c. **Trust Password:** The password that is associated with the Trust User ID on the remote system.
5. Click **OK**. The profile is added to the Block Storage Replication Page with an initial state of **Defined**.
6. Click the profile to validate and enable the profile. The status changes to **Enabled** as shown in Figure 4-2.

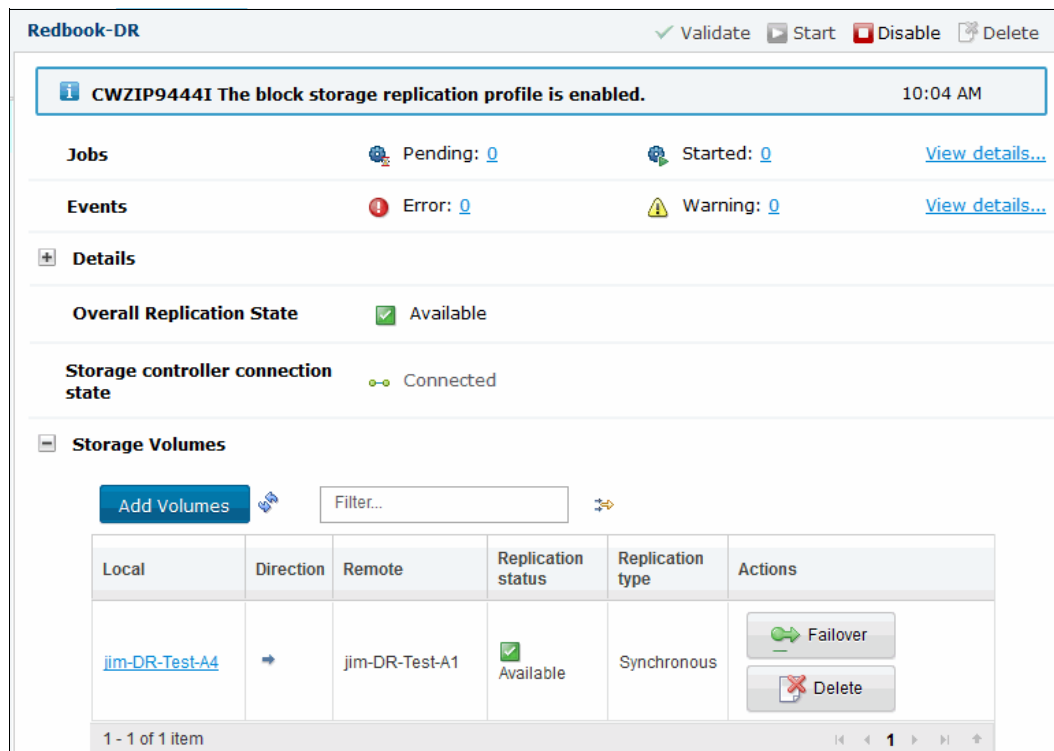


Figure 4-2 Block storage replication profile on the primary rack

To configure a block storage replication profile on the secondary rack, complete these steps:

1. From the System Console, select **System** → **Block Storage Replication**.
2. Click the **New** icon.

3. There are four required fields:
 - a. **Name:** Provide a descriptive name such as “Redbooks-DR.”
 - b. **Peer management location:** This is the management IP address (or FQDN) of the system to which a relationship is established (Primary rack)

Note: You can locate the management IP address for the Passive system by clicking **System** → **Network Configuration**. Expand the Management IP section. Use the floating IP address.

- c. **Trust User ID:** This user ID establishes the trust relationship between the host and remote system. The user must have full permissions to manage security.
 - d. **Trust Password:** The password that is associated with the Trust User ID on the remote system.
4. Click **OK**. The profile is added to the Block Storage Replication Page with an initial state of **Defined**.
5. Click the profile to validate and enable the profile. The status changes to **Enabled** as shown in Figure 4-3.

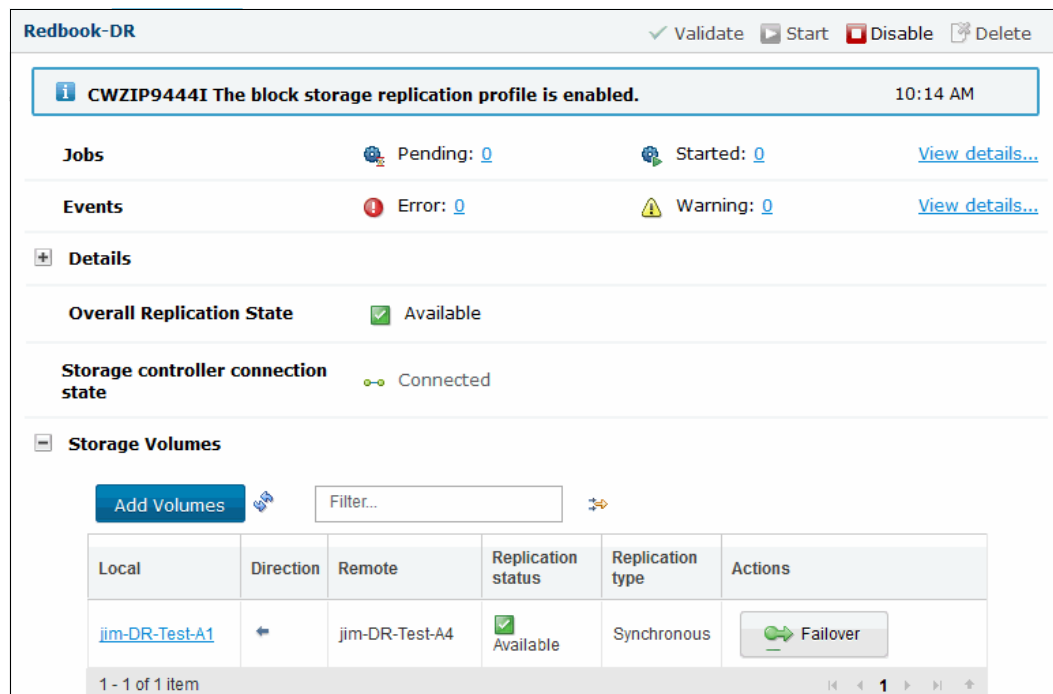


Figure 4-3 Block storage replication profile on the secondary rack

Now that block storage replication profiles exist on both the primary and secondary racks, you can add storage volumes for replication. This action links two storage volume pairs on the racks and starts the replication process.

1. On the primary rack, select **System** → **Block Storage Replication**.
2. Locate the appropriate block storage replication profile. Click **Add Volumes**.
3. Select the **Local Volume** (primary rack).
4. Select the **Remote Volume** (secondary rack). The interface only displays volumes of the same type and size.

5. Select the **Replication type** (either Synchronous or Asynchronous) as shown in Figure 4-4.

Note: Synchronous was selected for this book because the racks were in the same physical location.

Add storage volumes for replication

Local Volume: Remote Volume: Replication type: Synchronous Asynchronous

Add **Close**

Figure 4-4 Adding local and remote storage volumes for replication

6. Click **Add**.
7. Accept the replication request for each storage volume pair on the Passive rack.

Confirm that the storage volumes configured for replication on each rack are configured properly using these steps:

1. On the primary rack, select **System** → **Block Storage Replication**.
2. Select the block storage replication profile.
3. Expand **Storage Volumes**. You should see the local and remote volumes you selected earlier. Notice the arrow in the Direction field points to the storage volume on the secondary rack as shown in Figure 4-5.

| Local | Direction | Remote | Replication status | Replication type | Actions |
|--------------------------------|-----------|----------------|--------------------|------------------|--------------------|
| jim-DR-Test-A4 | → | jim-DR-Test-A1 | Available | Synchronous | Failover Delete |

1 - 1 of 1 item

Figure 4-5 Storage volume replication on primary rack

4. On the secondary rack, select **System** → **Block Storage Replication**.
5. Select the block storage replication profile.

- Expand **Storage Volumes**. You should see the local and remote volumes you selected earlier. Notice the arrow in the Direction field points to the storage volume on the primary rack as shown in Figure 4-6.

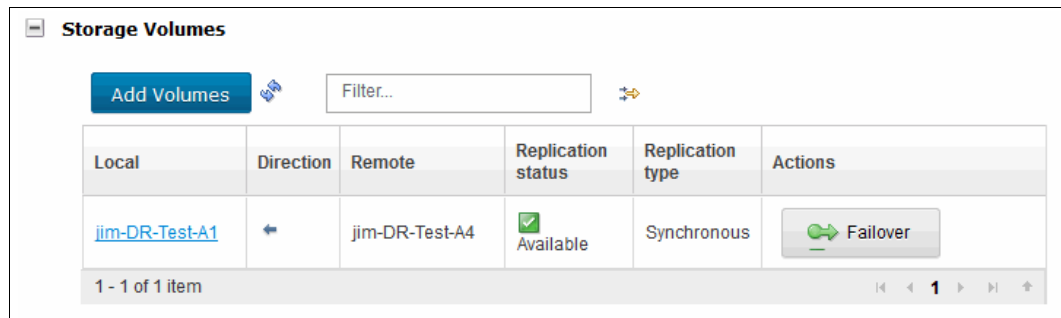


Figure 4-6 Storage volume replication on secondary rack

4.3 Configuring an Active/Active (Mirrored) GPFS deployment

For this book, two racks were used for this deployment. The first (primary) rack contains the GPFS Primary configuration. The GPFS Mirror configuration is on the other (secondary) rack. Normally a third rack would be used in this configuration for either the Mirror or the Tiebreaker.

The GPFS primary configuration on the first rack can support 1, 3, 5, or 7 GPFS server nodes. For this publication, one node was configured. The primary server can have either a mirrored or passive replica attached to it. The primary server cannot have both a mirrored and passive replica. If you attempt to deploy this configuration, you receive a message that indicates the configuration is not allowed.

The GPFS mirrored configuration consists of a GPFS mirror deployment and a GPFS tiebreaker deployment. These deployments are attached to the GPFS primary server deployed in the GPFS primary configuration.

Note: The configuration shown in this publication is not a typical Active/Active GPFS deployment. A typical configuration has each GPFS deployment on a separate rack. For example, GPFS Primary (rack1), GPFS Mirror (rack 2), and GPFS Tiebreaker on an iSCSI device.

4.3.1 Active/Active GPFS deployment: Steps

The following procedure shows the general steps to configure an active/active GPFS deployment. In this configuration, a GPFS Primary instance is deployed to one rack (primary rack), while a GPFS Mirror instance and GPFS Tiebreaker instance are deployed on another rack (secondary rack), possibly at a separate location. In general, to avoid latency problems with data transfer, this type of configuration is supported only for geographic distances less than 300 km apart.

To configure the primary rack, complete these steps:

1. On the primary rack, verify that a storage volume has been created. For more information, see 4.1, “Block storage configuration” on page 48. The storage volume should be configured as shown in Figure 4-7.

| | | | | | | | | |
|--------------------------|--------------------------------------|-------------------------------------|--------|--|-----------------|----------|--|---|
| <input type="checkbox"/> | RBHADR- GPFSPassiveConfig-Primary | <input checked="" type="checkbox"/> | 100 GB | RedBookHAC External CG | Block Shared | Internal | This is the Primary part of the passive Config with Rack 1 | 1 |
|--------------------------|--------------------------------------|-------------------------------------|--------|--|-----------------|----------|--|---|

Figure 4-7 Block storage configuration on primary rack

2. On the primary rack, create the GPFS Primary pattern by completing these steps:
 - a. From the Workload Console, select **Patterns** → **Virtual Applications**.
 - b. Select **GPFS Pattern Type 1.2**.
 - c. Click **Create New**.
 - d. There are four pattern options. Select **GPFS Primary** and click **Start Building**. A new browser tab opens with the Pattern Builder available.
 - e. Use these guidelines and the information in Figure 4-8 on page 55,
 - **Name:** Provide a name based on standards. This publication uses RBHADR_GPFS_Server.
 - **GPFS Managers Key:** Generated automatically. The key must be the same as the Mirror and Tie configuration. The GPFS Cluster Key and GPFS Clients Key do not need to be configured.
 - **GPFS Configurations:** Select Primary Configuration.
 - **Number of GPFS Node(s):** Select 1 node.
 - **Cluster Name:** Provide a cluster name. This publication used Primary_Cluster.
 - **File System name:** Enter file system name. This publication used RBHADRfileSystem.
 - Clear the **Active Configuration** and **Passive Configuration** check boxes.
 - f. Save the pattern with a standard name.

▼ **GPFS Server**

*** Name**

▼ **Key Management**

GPFS Managers Key

GPFS Cluster Key

GPFS Clients Key

▼ **GPFS Configurations**

Primary Configuration ▼

*** Number of GPFS Node(s)**

*** Cluster Name**

*** File System name for selected shared volumes**

*** Snapshot Interval (minutes)**

▶ Active Configuration (Mirror and Tie)

▶ Passive Configuration

Figure 4-8 Sample GPFS Primary server configuration

3. On the primary rack, deploy the Primary GPFS pattern by completing these steps:
 - a. Before deploying, verify that the component attributes and settings are the same as the pattern you just created. See Figure 4-9.

RBHADR GPFS Server

Fill in the required values for component **RBHADR GPFS Server** for this pattern.

▶ Key Management

▼ GPFS Configurations

Primary Configuration

* Number of GPFS Node(s) 1 node

* Cluster Name Primary_Cluster

* File System name for selected shared volumes RBHADRfileSystem

* Snapshot Interval (minutes) 60

▶ Active Configuration (Mirror and Tie)

▶ Passive Configuration

Figure 4-9 Primary GPFS pattern deployment window

- b. Click **Continue to Distribute**.
- c. You should see two components. The GPFS-Manager component should have a blue dot next to the VM. Move the cursor over the VM box. Click the **Edit (Pencil)** icon.

- d. Select the storage volume that you configured earlier. Figure 4-10 provides an example.

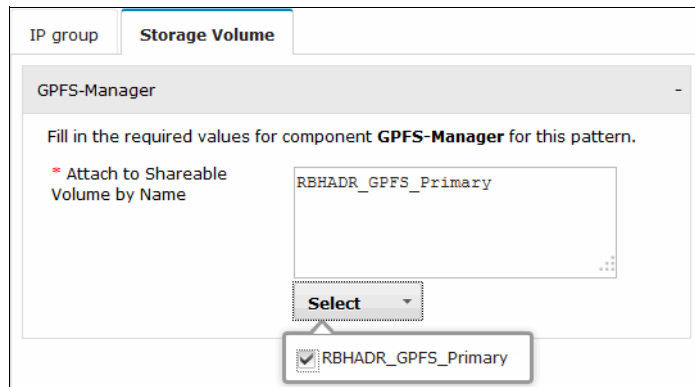


Figure 4-10 Add a storage volume to the Primary server configuration

- e. **Click Deploy.** Check the deployment status in the History section of the instance. The deployment takes 5-10 minutes depending on system resource availability.
- f. No configuration is necessary on GPFSServer-Main.

To configure the Secondary rack, complete these steps:

1. On the secondary rack, create a Mirror GPFS pattern. The configuration is similar to the primary pattern with a few key differences:
 - **Name:** Provide a unique name.
 - **GPFS Managers Key:** artifacts/manager_key_linux.tar.gz (this is the default value and should match the key in the primary configuration).
 - **GPFS Configurations:** Mirror Configuration.
 - **Number of GPFS Node(s):** 1 node (this value should match the primary configuration).
 - **File System name for selected shared volumes:** This value should match the primary configuration file system name.
2. Save the pattern.
3. Create a Tiebreaker GPFS pattern. The configuration is similar to the primary pattern with a few key differences:
 - **Name:** Provide a unique name.
 - **GPFS Managers Key:** artifacts/manager_key_linux.tar.gz (this is the default value and should match the key in the primary configuration).
 - **GPFS Configurations:** Tiebreaker.
 - **File System name for selected shared volumes:** This value should match the primary configuration file system name.
4. Save the pattern.

5. Deploy both the Mirror and Tiebreaker GPFS patterns. Be sure to attach the appropriate block shared storage to each deployment as shown in Figure 4-11.

| | | | | | | |
|--------------------------|------------------------------------|-------------------------------------|--------|-------------------------------------|--------------|----------|
| <input type="checkbox"/> | RBHADR_GPFS_Mirror | <input checked="" type="checkbox"/> | 100 GB | AustinRack Internal | Block Shared | Internal |
| <input type="checkbox"/> | RBHADR_GPFS_Tie | <input checked="" type="checkbox"/> | 20 GB | AustinRack Internal | Block Shared | Internal |

Figure 4-11 Sample block shared volumes for mirror and tie configuration

6. Make a note of the GPFS-Manager IP address for each deployment as shown in Figure 4-12 and Figure 4-13. These are added to the primary configuration by the administrator in the next step.

| | | | | | | |
|-------------------------------------|---|-------------|---|--|--|----------|
| <input checked="" type="checkbox"/> | GPFS-Manager.11411493797964 | 9.3.169.116 | <input checked="" type="checkbox"/> Running | <div style="width: 48%; background-color: green;"></div> 48% | <div style="width: 48%; background-color: green;"></div> 48% | Manage ▾ |
| <input checked="" type="checkbox"/> | GPFS-Server-Main.11411493797965 | 9.3.169.117 | <input checked="" type="checkbox"/> Running | <div style="width: 100%; background-color: red;"></div> 100% | <div style="width: 54%; background-color: green;"></div> 54% | Manage ▾ |

Figure 4-12 IP address for the GPFS Mirror server

| | | | | | | |
|-------------------------------------|--|-------------|---|--|--|----------|
| <input checked="" type="checkbox"/> | GPFS-ManagerTie.11411494552461 | 9.3.169.118 | <input checked="" type="checkbox"/> Running | <div style="width: 100%; background-color: red;"></div> 100% | <div style="width: 62%; background-color: green;"></div> 62% | Manage ▾ |
| <input checked="" type="checkbox"/> | GPFS-Tiebreaker.11411494552462 | 9.3.169.119 | <input checked="" type="checkbox"/> Running | <div style="width: 99%; background-color: red;"></div> 99% | <div style="width: 62%; background-color: green;"></div> 62% | Manage ▾ |

Figure 4-13 IP address for the GPFS Tiebreaker server

To configure the Primary rack, complete these steps:

1. On the primary rack, add the Mirror GPFS server to the Primary GPFS instance. The dialog is shown in Figure 4-14 on page 59.
 - a. From the Workload Console, select **Instances** → **Virtual Applications**. Select the Primary Server instance.
 - b. Select **Manage**. This action opens the Instance Console.
 - c. Select the GPFS-Manager (not the GPFS-Server-Main).
 - d. Select **Operations**.
 - e. In the Operations window, select **GPFS_Manager**.
 - f. Expand **Add New Member**.
 - **Member Type:** Mirror
 - **Manager IP:** <enter the IP of the Mirror GPFS Server>
 - g. Click **Submit**.

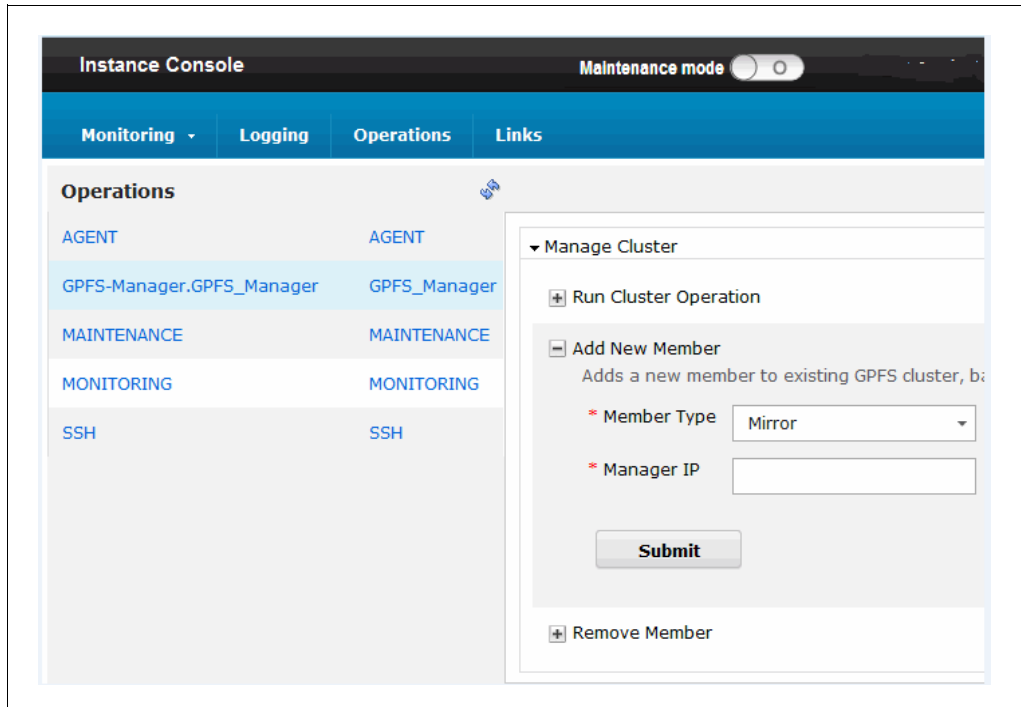


Figure 4-14 Add a Mirror to Primary cluster

- h. Confirm that you want to add the cluster member.
- i. Verify that the operation was successful. Look in the Operation Execution Results window. You should see a message similar to Figure 4-15.

| Name | Status | Created Time | Result | Return Value |
|----------------|--------|--------------------------|---|---|
| Add New Member | Done | Sep 23, 2014, 1:10:43 PM | GPFs-Manager.11411480280858.GPFs_Manager: Success | GPFs-Manager.11411480280858.GPFs has been invoked. Click here to view the |

Figure 4-15 Operation Execution Results

Note: In this configuration, the Mirror server was added to the Primary server after the Primary server was deployed. Optionally, you can add the Mirror server to the Primary server configuration before deploying the Primary server.

2. On the primary rack, add the Tiebreaker GPFS instance to Primary GPFS instance. The procedure is similar to adding the Mirror to the Primary.
 - a. In the Operations window, select GPFS_Manager.
 - b. Expand **Add New Member**.
 - **Member Type:** Tiebreaker
 - **Manager IP:** <enter the IP of the Tiebreaker GPFS Server>
 - c. Click **Submit**.
 - d. Confirm that you want to add the cluster member.
 - e. Verify that the operation was successful. Look in the Operation Execution Results window. You should see a message that indicates a successful operation.

Note: In this configuration, the Tiebreaker server was added to the Primary server after the Primary server was deployed. Optionally, you can add the Tiebreaker server to the Primary server configuration before deploying the Primary server.

3. On the primary rack, verify the GPFS cluster configuration by running two separate cluster operations:
 - a. Get Configuration Type as shown in Figure 4-16.
 - i. In the Operations window, select **GPFS_Manager**.
 - ii. Expand **Manage Cluster** → **Run Cluster Operation**.
 - iii. Select **Get Configuration Type**.
 - iv. Click **Submit**.

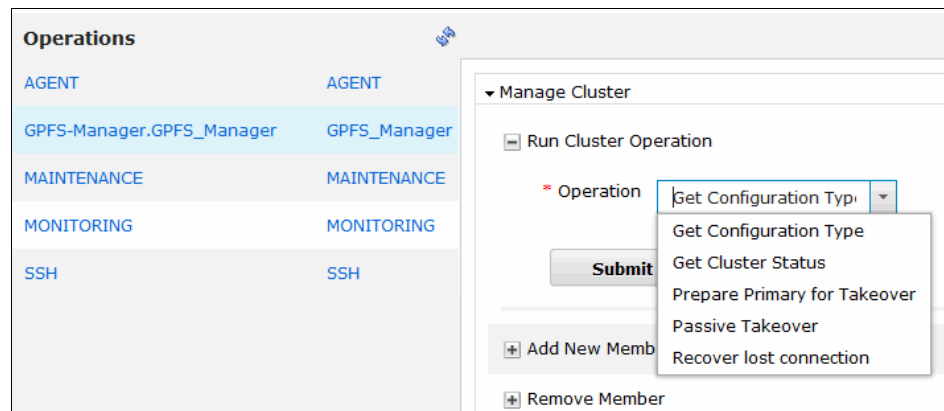


Figure 4-16 Get Configuration Type operation

- v. In the Operation Execution Results window, you should see a message indicating that the operation was successful. In the Return Value column, click the link. You should see a message similar to Figure 4-17.

```

2014-09-23 18:55:51.269620
Configuration Type : Primary with 1 gpfs node(s)
Cluster Name: Primary_Cluster
Rack Name: PrimRack
Failure Group: 1
Mirror Manager IP: 9.3.169.116
Tie Manager IP: 9.3.169.118
  
```

Figure 4-17 Results of Get Configuration Type operation

- b. Get Cluster Status:
 - i. In the Operations window, select **GPFS_Manager**.
 - ii. Expand **Manage Cluster** → **Run Cluster Operation**.
 - i. Select **Get Cluster Status**.
 - ii. Click **Submit**.
 - iii. In the Operation Execution Results window, you should see a message indicating that the operation was successful. In the Return Value column, click the link. You should see a lengthy status report.

4.4 Configuring an Active/Passive GPFS deployment

In an active/passive GPFS configuration, you deploy a GPFS Primary instance and a GPFS Passive instance on separate racks. This deployment is used by several of the scenarios that are described in this publication.

4.4.1 Active/Passive GPFS deployment: steps

The following are the required steps for an Active/Passive GHPFS deployment:

1. On the primary rack, follow the configuration steps 1 - 4 outlined in 4.3.1, “Active/Active GPFS deployment: Steps” on page 53 to deploy the GPFS Primary server. The deployment steps for the GPFS Primary server are the same in both deployments.
2. Verify that the appropriate storage volumes have been created. The storage volumes must be identical in size and number to those on the primary rack. On Intel-based systems, use block shared storage. On Power-based systems, use block storage. This configuration used 100 GB of block storage. This storage volume must match the size and type on the primary rack. See Table 4-2 on page 48 for storage volume specifications on the secondary rack. An example of the storage volume used for the passive configuration on the secondary rack is shown in Figure 4-18.

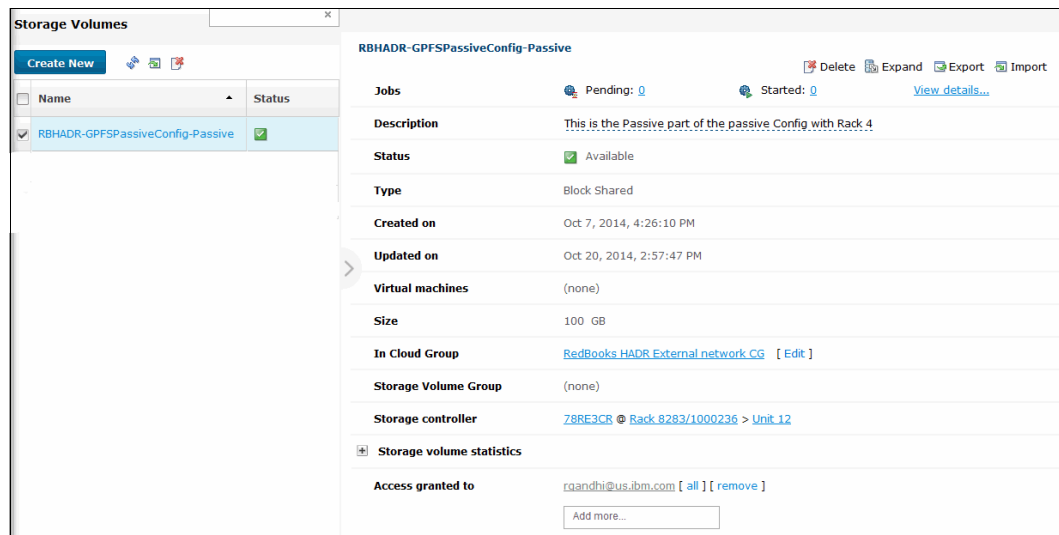


Figure 4-18 Storage volume configuration on Passive deployment.

3. On the secondary rack, create GPFS Passive pattern. The configuration is similar to the primary pattern, except the GPFS configuration is “Passive Configuration”. See Figure 4-19 on page 62 for a completed configuration.
 - **Name:** Provide a name based on standards. This publication uses GPFS_Passive.
 - **GPFS Managers Key:** Generated automatically. The key must be the same as the Primary configuration. The GPFS Cluster Key and GPFS Clients Key do not need to be configured.
 - **GPFS Configurations:** Select Passive Configuration.
 - **Number of GPFS Node(s):** Select 1 node.

- **Cluster Name:** Provide a cluster name. This publication used *Primary_Cluster*.
- **File System name:** Enter file system name. This publication used *RBHADRfileSystem*.

The screenshot shows the configuration interface for a GPFS Server. It includes the following fields and options:

- Name:** A text input field containing "GPFS Passive".
- Key Management:** A checked checkbox with a dropdown arrow.
- GPFS Managers Key:** A text input field containing "artifacts/manager_key_" with "Browse" and "Delete" buttons.
- GPFS Cluster Key:** A text input field with a "Browse" button.
- GPFS Clients Key:** A text input field with a "Browse" button.
- GPFS Configurations:** A dropdown menu showing "Passive Configuration".
- Number of GPFS Node(s):** A dropdown menu showing "1 node".
- Cluster Name:** A text input field containing "Cluster_Name".
- File System name for selected shared volumes:** A text input field containing "RBHADRfileSystem".

Figure 4-19 GPFS Passive configuration

4. Save the pattern.
5. On the secondary rack, deploy the GPFS Passive pattern.
 - a. Before deploying, verify that the component attributes and settings are the same as the pattern you just created.
 - b. Click **Continue to Distribute**.
 - c. You should see two components. The GPFS-Manager component has a blue dot next to the VM. Move the cursor over the VM box. Click the **Edit (Pencil)** icon.
 - d. Select the storage volume that you configured earlier.
6. Attach the replicated storage volumes.

4.4.2 Active/Passive setup and takeover

This setup uses the GPFS Passive configuration, where the GPFS Primary server is deployed on the primary data center (PDC-1) and the GPFS Passive server is deployed on the secondary data center (SDC-1). As shown previously, in this configuration, both servers use block storage. A replication pair is created between the two block storage volumes of the GPFS server.

During the passive takeover, several manual steps are performed. The high-level steps that are listed here are for information only:

1. On GPFS Primary server, select **Prepare Primary for Takeover**.
2. Perform manual **Failover** operation on the Passive Block Storage volume. This action deletes the replication between the two block storage volumes.
3. On DR rack, for the GPFS Passive server, select **Passive Takeover**. This action makes the GPFS Passive server the primary server.
4. On DR rack, redeploy GPFS Shared Service to point to the new GPFS Primary.

4.5 Deploy GPFS Shared Service

In order for a GPFS client to communicate with its GPFS server, it must supply an IP address or communication key. To avoid manually configuring this information for each client, you can deploy a **GPFS shared service**. The GPFS shared service can be configured with the IP address or communication key. Deploy instances of this shared service in the same cloud groups where future GPFS client deployments are expected. The shared service does not need to reside in the same cloud group, or even the same PureApplication System rack, with the GPFS server.

Note: GPFS Clients can also be configured to point to an external GPFS server that is deployed outside of a PureApplication System. At deployment time, the GPFS Client provides the IP address for the main external GPFS server, and runs a set of script packages to set up communication with the external GPFS server.

The following steps outline the procedure to deploy the GPFS shared service:

1. From the Workload Console, select **Instances** → **Shared Services** → **Create New**.
2. Select **IBM Shared Service for GPFS (External) 1.2.0.0**. See Figure 4-20.

Deploy Shared Service
Select a shared service instance from the list and deploy it into the cloud.

| | |
|---|---|
| Caching Service 2.1.0.0 | Application name: IBM Shared Service for GPFS |
| Database Performance Monitoring 1.0.2.0 | Application ID: a-904d567f-8429-4a70-8fd3-90cd08b112c6 |
| ELB Proxy Service 2.1.0.0 | Created by: admin |
| IBM Endpoint Manager Service 1.0.1.0 | Created on: Oct 6, 2014, 4:34:18 PM |
| IBM Shared Service for GPFS (External) 1.2.0.0 | Updated by: admin |
| Red Hat Satellite Service (External) 1.0 | Updated on: Oct 6, 2014, 4:34:18 PM |
| Red Hat Update Infrastructure(RHUI) Service 1.0 | Pattern Type: gpfs |
| | Version: 1.2 |
| | Description: Provides access to a GPFS Cluster Manager where the workload requests a highly available file system. |

OK Cancel

Figure 4-20 GPFS Shared Service

3. Get the client key from the GPFS-Manager server.
 - a. Select **Instances** → **Virtual Applications** → **<Instance name>**.
 - b. Click **Manage**.
 - c. Click the Operations tab.
 - d. Select **GPFS-Manager.GPFS_Manager**.
 - e. Expand **Manage Keys** → **Retrieve Keys**.
 - f. Select **Client**.
 - g. Click **Submit**. After the operation finishes, in the **Return Value** field, click **client private key**. The client private key should look similar to Figure 4-21.

```

172.21.20.48 ____-----BEGIN RSA PRIVATE KEY-----
MIIIEogIBAAKCAQEAAw5ByCH5GThrTvonljm8UqS1SgGanXxoDoQUNUrMX8IgrDaXz
6LWwqoRyr/1v1rRC46C5BXUGYtxLIYzWqzPro2AuKcqqRUHPIaj3gW/FFmus24CB
25ub31go6Jv3KEVtUqQhU08x4pc8B6B0eYDLWRyOqtd0VmRdTitf3oIMc/FLRNj
XoJ18tI162thns8+cidZ/N86KYTKz8V2JS8evFE/IasSp3Dlyq4qlihYkvi8R715
EUoAJFZ7hcRH6Y+uuD60dgXTCCa8PGMSWI54Q2t8B+BF9hIYFFkaWBfJ0F5M5Dp
77/B5Qqb/M+CUd282Vrkoibgar1zdkI8/ve8PQIBiWKAQEAss01DxRdiT0ZXb/n
0q6zzdyjWCNXXmDtYASed38r1JnA9ok2hE4d3brsg6I6I2L7U8Y1gVUNJy/P012Y
Au2kPZnDznAmpbiC3PJBYGYwiYcEcoQe8QNMrvGEfOZeULRpylLNRqRU07G0DrzN
68r71zdD3d6BqXZ2XIuXI/fM+1X2jJ1/aExtGdmSI49/rqQqcTck2DwpFps1K0fB
8bRyZbradwXWS/o32v0wSghW1pjmNR8GEXvBFSGqzE5KTLiRGBelcS8YTcs080Mk
UERln9EKMAZyICm8CoQSSoXNWNLANGQJ5CJWv3uPexGKWS3T+BLE2WXVveAgcz1+
N5quCwKBgQDhVFNjphx5PUz1mr8neF/a+7Sc4qS+ZMKb4saLxQOo0318Mormu/Yg
1y/kz9myOVwy+9CQOTdW2FJk4Wcy53TdmZXX3TSAzuYbyedqBH1Kf1WIhttnDR3H
foweBk2U2QnwH0jvp5geFD/T8xMLkZau9jHSSDM1181sB8vjUt2a1wKBgQDeL108
X1J1YT84ecdGf2vVCPnGzbiu1A1BjahuP84Yy2Nj/bHJ1GEX2/1Y9VNHXUvNqdL5
nPN0H0dD1OdzvjDybE7uJQXQCvq9P5GyjRoOuut11L0NN4zbNxbSS7Bxs0oGaTqP
dqIU71kjGyFHBzRzc9RunVxJC0Lj8GeXn0jzCwKBgE1BsoiQuU4jpVuFg139YrF6
3ta0Gzn2qR+A852ioitBMPdhh2UNPnGo3T1BxvtG3chWViMF5x22DZ7uMgLTA38t
V/I9NpKQFGFMids0vT4dB2ICWdowYftBVJyUcl7rU91MjgkGQsh7+KBwNDEM6mz
fsvA+51YnjOq5tGYwQHjAoGAOSHbI1IYz2DdOwdTKRzL3fBJH4K6yCUeRXMOZtg
4clvCxVZmjKDBiKfNCHURh+3TnHXKj5NMV/fjeltRdYitgCp6skQIaxjEIgkINTo5
5ofWIX/Y14NQgxzLWqW/qDVqznLU10pkMUTcaBarTOGLGKP0y/yUEshhQFPDxD7m
32kCgYEAj59fcvrVIZBS/H1PPCM/KGFvPJNFk89dqT0Qa4AayFEIqBkzSQxsKv7R
XTgwaFpScMnv6YT2nAys5ATk1BiXfcvHLo62PeSIPrJnpwXJ0Qh3AFD58xEdimob
iqieytnJvZsItz0kdy1JblhOk12aoo64tedTf2fSCaE9t8DcAYQ=
-----END RSA PRIVATE KEY-----

```

Figure 4-21 Manager IP and private key file

- h. Cut and paste the client key into **GPFS Manager IP and Client Key**.

4.6 External storage configuration

External storage was not configured for this book. Information about configuring external storage is found in the “Configuring the system to use external storage devices” at:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extst_orage_cfg.dita

4.7 Network configuration and cloud resources configuration

This section provides a description of the network and cloud configuration for this publication. The descriptions provided are for illustrative and guidance purposes only. Any naming conventions and configurations are specific to the systems used for this project.

Perform networking and cloud configuration before any of the HADR scenarios are implemented. See Chapter 3, “High availability and disaster recovery scenarios” on page 29 for a list of scenarios.

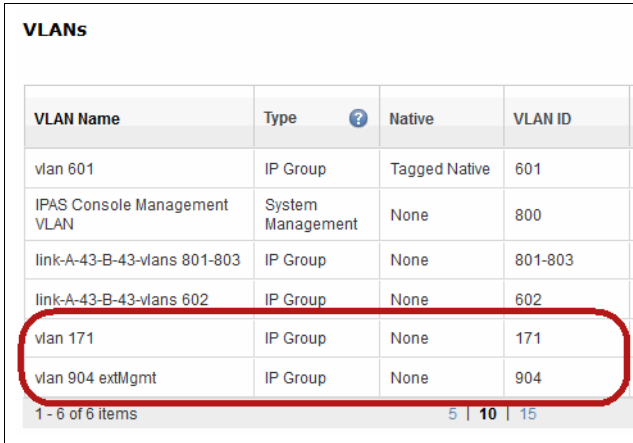
4.7.1 Network configuration

Configuring the network is dependent upon the scenarios you decide to implement. Some scenarios consume more network resources than others. For example, scenario *WAS_1: WebSphere cell in the same rack (PDC-1 with transactions in GPFS)* requires far fewer IP addresses than scenarios that implement the Business Process Manager pattern.

VLANS

A virtual local area network (VLAN) logically isolates its network traffic from that of other VLANs on the same network as a separate broadcast domain. This means that the network traffic of applications using two IP groups with different VLAN IDs transmits on (seemingly) independent networks. PureApplication System does not define VLANs, which are defined in the network by the network administrator. However, it does use VLANs extensively. PureApplication System uses VLANs in two distinct ways: As a management VLAN or an application VLAN.

Two VLANs were used. The VLAN configuration on the primary and secondary racks was the same. One VLAN was used for data traffic and the other was used for cloud management. The two VLANs configured are shown in Figure 4-22.



| VLAN Name | Type | Native | VLAN ID |
|------------------------------|-------------------|---------------|---------|
| vlan 601 | IP Group | Tagged Native | 601 |
| IPAS Console Management VLAN | System Management | None | 800 |
| link-A-43-B-43-vlans 801-803 | IP Group | None | 801-803 |
| link-A-43-B-43-vlans 602 | IP Group | None | 602 |
| vlan 171 | IP Group | None | 171 |
| vlan 904 extMgmt | IP Group | None | 904 |

1 - 6 of 6 items 5 | 10 | 15

Figure 4-22 Sample VLAN configurations

4.7.2 Cloud resources configuration

Generally, cloud configuration in PureApplication System involves configuring IP groups, cloud groups, compute nodes, and environment profiles. Configuration establishes relationships among these resources. The relationship between these resources is shown in Figure 4-23.

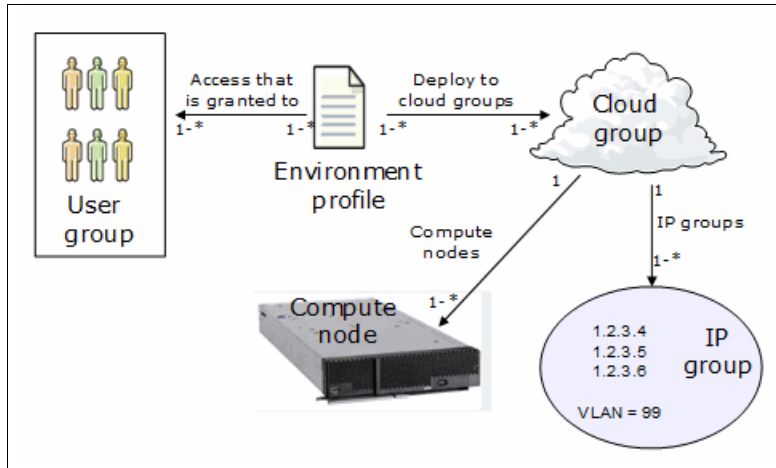


Figure 4-23 Relationship of cloud resources in PureApplication System

Compute nodes

A compute node is essentially a compact computer, rather like a blade server. As a computer, a compute node can run an operating system. However, in PureApplication System, instead of a traditional operating system, each compute node runs a hypervisor.

The Intel-based systems used in this publication were both “mini” systems with four compute nodes on each rack. An example rack configuration is shown in Figure 4-24. The primary (PDC-1) and secondary (SDC-2) each had the same number of compute nodes.

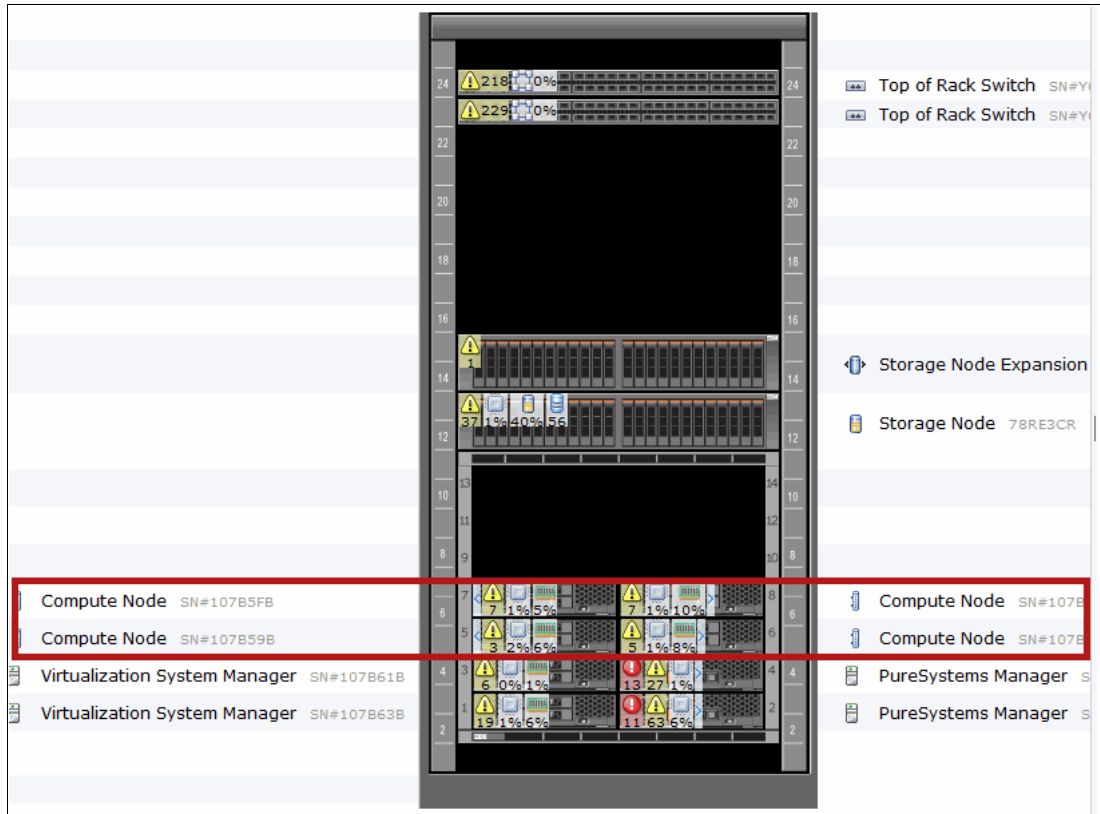


Figure 4-24 Example of compute nodes on “mini” system

IP groups

An IP group is a set of IP addresses, expressed either as a list or a range. Because a network does not work properly with duplicate IP addresses, each address in the group must be unique and an address can belong to only one group. The group also has settings for the ID of the VLAN to use for communication, and settings for how to connect to the external network that the VLAN is part of. All of the addresses in the set must belong to the same subnet of the external network, which is the one indicated by the netmask.

Four IP groups were configured on each of the racks. Each configuration was similar to the one shown in the cloud group configuration in Figure 4-25 on page 69. Two IP groups were used for data management (shown as associated with VLAN ID 171) and two IP groups were used for cloud management. (shown as associated with VLAN ID 904).

The configurations in this Redbooks publication consumed a large number of IP addresses. Before implementing these scenarios, work with your network administrator to reserve an appropriate number of IP addresses. The IP group configuration is shown in Table 4-3.

Table 4-3 Sample IP group configuration

| IP group name | VLAN | Used for | Number of IP addresses |
|-------------------------|------|------------------|------------------------|
| R4-v904-Data-10.14.1.x | 904 | Data | 93 |
| R4-v904-extMgmt | 904 | Cloud Management | 249 |
| R4-vlan171-1.externMgmt | 171 | Cloud Management | 63 |
| R4-vlan171-2 | 171 | Data | 93 |

Cloud groups

A cloud group is a virtualized platform for running workloads, and acts like a logical computer. You can also think of a cloud group as a cloud within a cloud, with PureApplication System being the larger cloud. It accomplishes two main goals:

- ▶ System segmentation: It divides a PureApplication System into one or more logical computers. Groups run isolated from each other.
- ▶ Compute node aggregation: It groups one or more compute nodes along with at least one IP group into a logical computer that can have greater capacity than a single node.

A cloud group acts like a logical computer that is a virtualized platform. Here are some rules that affect cloud groups:

- ▶ Each virtual machine that runs in a cloud group runs in one of the cloud group's compute nodes and runs with an IP address from one of the IP groups.
- ▶ A particular compute node can belong to only one cloud group (at most). Typically, a cloud group contains at least two compute nodes so that the group can keep running even if one of the nodes fails. This configuration limits the number of cloud groups that a single PureApplication System can support.
- ▶ A particular IP group can belong to only one cloud group (at most). Each cloud group also requires its own management VLAN.
- ▶ Cloud groups create isolated runtime environments, such that workloads running in one group are not affected by workloads running in another group.

The cloud group configuration for this publication was *externally managed* with four IP groups and two compute nodes as shown in Figure 4-25. This configuration was done by setting **Cloud Management** to **By way of external network**. An externally managed cloud group connects the management traffic of a virtual machine to an external VLAN, allowing the traffic to be subject to external routing and firewalls when that traffic crosses VLANs. It also allows virtual machines that are deployed to this cloud group to communicate with virtual machines in other cloud groups or other systems for management and monitoring.

RedBookHADR External CG

 Delete
 Refresh

Type

Average

Reserve resources for availability

Yes ▼

High availability

Active

Cloud Management

By way of external network

IP Groups

1 - 4 of 4 items

| <input type="checkbox"/> | Status | IP Groups | Used For | VLAN | Action |
|--------------------------|-------------|-------------------------|------------------|------|--------|
| <input type="checkbox"/> | ✓ Available | R4-vlan171-2 | Data | 171 | |
| <input type="checkbox"/> | ✓ Available | R4-v904-extMgmt | Cloud Management | 904 | |
| <input type="checkbox"/> | ✓ Available | R4-vlan171-3.externMgmt | Cloud Management | 171 | |
| <input type="checkbox"/> | ✓ Available | R4-v904-Data-10.14.1.x | Data | 904 | |

Compute nodes

Total: 2 ✓ Available: 2

| Status | Power status | Compute nodes | Physical CPU utilization | Physical memory utilization | Action |
|-------------|--------------|---|---|---|--------|
| ✓ Available | ⊕ Powered On | SN#106625B @ Rack 8283/1000202 > Chassis 1 > Node Bay 5 | <div style="width: 100%; height: 10px; background-color: #ccc; position: relative;"><div style="width: 0%;"></div></div> 0% | <div style="width: 100%; height: 10px; background-color: #ccc; position: relative;"><div style="width: 8%;"></div></div> 8% | |
| ✓ Available | ⊕ Powered On | SN#106624B @ Rack 8283/1000202 > Chassis 1 > Node Bay 6 | <div style="width: 100%; height: 10px; background-color: #ccc; position: relative;"><div style="width: 1%;"></div></div> 1% | <div style="width: 100%; height: 10px; background-color: #ccc; position: relative;"><div style="width: 5%;"></div></div> 5% | |

1 - 2 of 2 items

Figure 4-25 Cloud group configuration

Because the cloud group is externally managed, you need to configure extra IP addresses. The IP addresses that you specify provide workloads in cloud groups managed by way of external network access to the workload console and system console. The IP addresses you specify are for the local system, not the remote system. The configuration is available from the System Console by selecting **System** → **Network Configuration** → **Additional IPs for Cloud Management by way of External Networks**. See Figure 4-26 for an example configuration.

Optional

Additional IPs for Cloud Management by way of External Networks

These IP addresses provide workloads in cloud groups managed by way of external networks access to the workload console.

VLAN: ?

Configure IPv4 addresses ?

Specify the IP addresses to route all network traffic for a particular cloud group outside the system. Proper operation of this configuration requires access to certain ports. For more information on which ports, see the planning details in the documentation.

* IP address 1 (system console access):

* IP address 2 (workload console access):

* Netmask:

* Gateway:

Figure 4-26 Additional IP configuration for externally managed cloud groups

Environment profiles

An environment profile defines policies for how patterns are deployed to one or more cloud groups and how their instances run in the cloud group. To deploy a pattern, a user selects a profile for running the deployment, which in turn specifies the cloud groups the deployer can deploy patterns to. The deployer should think of the environment profile as the deployment target.

For this publication, two environment profiles were created on each system. One profile was configured for an internally managed network and the other profile used an externally managed network. These configurations mean that the environment profile was associated with an environment profile with **Cloud Management** set to **By way of internal network** or **By way of external network**. Figure 4-27 shows an environment profile with associated externally managed cloud groups.

RedBook-External-R1R4

Description: None provided

Hypervisor type: PureSystems_ESX

Environment profile type: All

Network: Externally Managed

Created on: Oct 2, 2014, 4:42:05 PM

Current status: ✓ Environment profile can now be used for deployments

Updated on: Oct 24, 2014, 5:47:31 PM

Virtual machine name format: None provided

IP addresses provided by: IP Groups

Deployment priority: Platinum - High(16) Medium(8) Low(4)

Time zone: Default time zone configured for the virtual image

Language: Default language configured for the virtual image

Deploy to cloud groups:

| Name | Type | Location | Alias |
|-----------------------------------|--------|----------|----------------------------------|
| RedBookHADR External CG | Public | 1000202 | RedBookHADR External CG [remove] |
| RedBooks HADR External network CG | Public | 1000236 | RedBooks HADR External [remove] |

Figure 4-27 Sample environment profile

4.8 Multisystem environment deployment

A multisystem environment consists of two or more systems that are connected to each other physically (by networking) and logically (in a defined relationship). The systems in a multisystem environment might be at the same or different geographic locations.

Besides being connected to one another, each system in this environment might have several local and remote devices connected to it. In a multisystem environment, because systems are interconnected, data can be moved through and between them, from any system to any system. Similar to a single system environment, in a multisystem environment you continue to access the resources in your cloud group and manage multiple environments from a single console.

Note: Multisystem environment deployment is also referred to as “multi-rack” or “multi-target” deployment. The official IBM name for this feature is multisystem environment deployment.

PureApplication System provides continuous availability for key applications by deploying across multiple systems in a multisystem environment. The multisystem environment contains *management domains* and *deployment subdomains*. Systems that are part of a multisystem environment are also referred to as *locations*.

You can use a multisystem environment to perform a number of tasks:

- ▶ Deploy virtual machines across multiple locations in a region.
- ▶ Determine whether application content across multiple system locations and data centers is synchronized.
- ▶ Deploy patterns across multiple locations in a deployment subdomain.
- ▶ Manage catalog artifacts at different levels of granularity, including details for a single item, across a management domain.
- ▶ Copy virtual system images from one system location to another within the management domain.
- ▶ Edit script packages in the catalog and synchronize the changes to all system locations in the management domain.

4.8.1 Management domains

A management domain is used to create a management relationship between two or more PureApplication Systems. The systems associated with the management domain are called *locations*. Locations included in the management domain can be geographically separated by any distance. The systems, or locations, in a management domain can share patterns, users, and other catalog content, which you can manage through a single common console interface.

Systems in a management domain must share a common LDAP user repository, which provides a common set of users across systems. User permissions must be set explicitly on each system.

IP network connectivity is required between all systems. There is no limit on the number of systems in a management domain. There is no limit on distance between systems in a management domain.

4.8.2 Deployment subdomains

Up to two locations in the management domain can be further grouped into a deployment subdomain. Locations in the same deployment subdomain add the additional capability to share externally managed environment profiles and deploy multisystem pattern instances. Membership in a deployment subdomain is limited to two systems. You can define any number of deployment subdomains within a management domain.

To participate in a deployment subdomain, systems must be linked by a low-latency network (generally geographically <300 km apart). Each system in a deployment subdomain must have a minimum of 512 GB of storage available.

There is a requirement of a 1 GB iSCSI target that can be used as a tiebreaker if the systems cannot contact each other.

As always, systems can be stand-alone systems and not belong to any management domain or deployment subdomain.

4.8.3 Additional requirements

There are several additional requirements regarding management domains and deployment subdomains:

- ▶ You can remove a deployment subdomain from a management domain, but you cannot remove a system location from within a deployment subdomain.
- ▶ Removing a deployment subdomain also removes any externally managed deployments and environment profiles.
- ▶ When you remove a deployment subdomain, any locations that were part of that deployment subdomain are returned to the pool of available locations to be added to another deployment subdomain as needed.
- ▶ If a location is not a member of a deployment subdomain, you can remove the location from the management domain as needed.

4.8.4 System configuration

There are several steps that you need to complete to configure systems for multisystem deployment:

1. Configure IP addresses for external management.
2. Create a management domain.
3. Add locations to the management domain.
4. Create one or more deployment subdomains.
5. Add systems (locations) to a deployment subdomain.
6. Configure an iSCSI tiebreaker.

Configure IP addresses for external management

External management traffic requires an additional IP address for external system console access. The external network must enable routes that connect these additional IP addresses and the cloud management IP addresses. From the system console, click **System** → **Network Configuration** → **Additional IPs for Cloud Management by way of External Networks**. For Intel-based systems, configure **IP address 2** for workload console access as shown in Figure 4-28.

Optional

Additional IPs for Cloud Management by way of External Networks

These IP addresses provide workloads in cloud groups managed by way of external networks access to the workload console.

VLAN: ?

Configure IPv4 addresses ?

Specify the IP addresses to route all network traffic for a particular cloud group outside the system. Proper operation of this configuration requires access to certain ports. For more information on which ports, see the planning details in the documentation.

* IP address 1 (system console access):

* IP address 2 (workload console access):

* Netmask:

* Gateway:

Figure 4-28 Additional IP address configuration for cloud management by way of external networks

For Power-based systems, an additional IP address, IP address 3 (not shown) is needed for VM management.

Create a management domain

To create a management domain, complete these steps:

1. From the system console, select **System** → **Management Domain Configuration**.
2. Click **Create Management Domain**.
3. Enter a name.
4. Click **OK**.

In the example shown in Figure 4-29, **RedBooks HADR MR** is the name of the management domain.

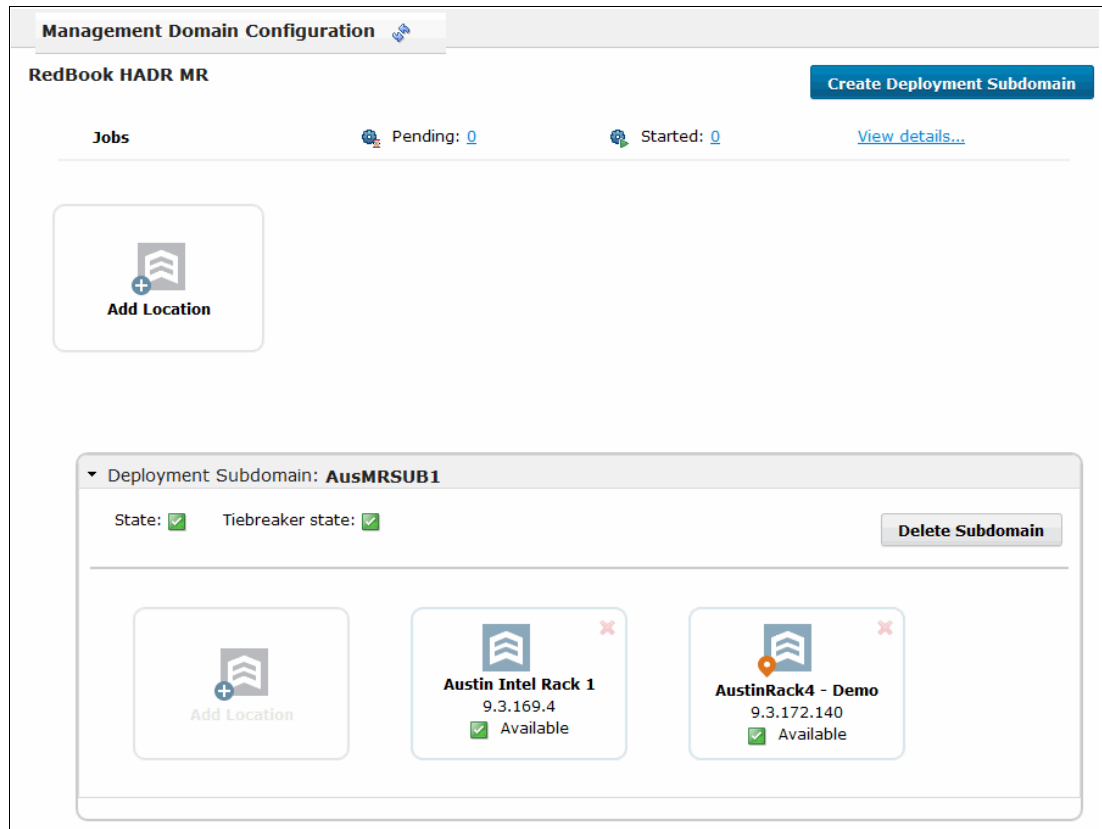


Figure 4-29 Management domain

Add locations to the management domain

When you add a location (system) to the management domain, a trust relationship is created between the systems that use OAuth. To add a location, the user must have the following permissions:

- ▶ Manage hardware resources (full permissions)
- ▶ Manage security (full permissions)

Click **Add Location** and provide the following (see Figure 4-30):

- ▶ Peer management location: Target system's management IP address.
- ▶ Trust user ID: User ID to log in to the target system and establish trust.

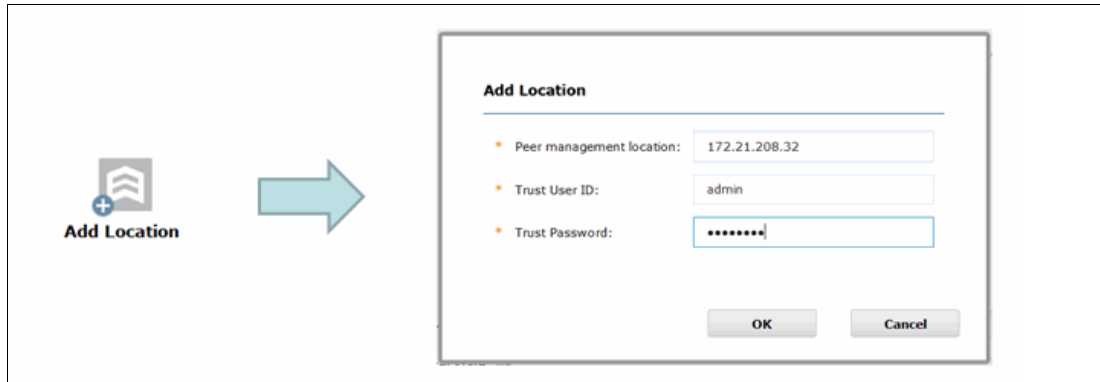


Figure 4-30 Add location to a management domain

4.8.5 Create one or more deployment subdomains

You can create a deployment subdomain after the management domain has been created. From the system console, select the management subdomain. Click **Create Deployment Subdomain**. Provide the name of the deployment subdomain.

Figure 4-31 shows a deployment subdomain that was configured for this publication.

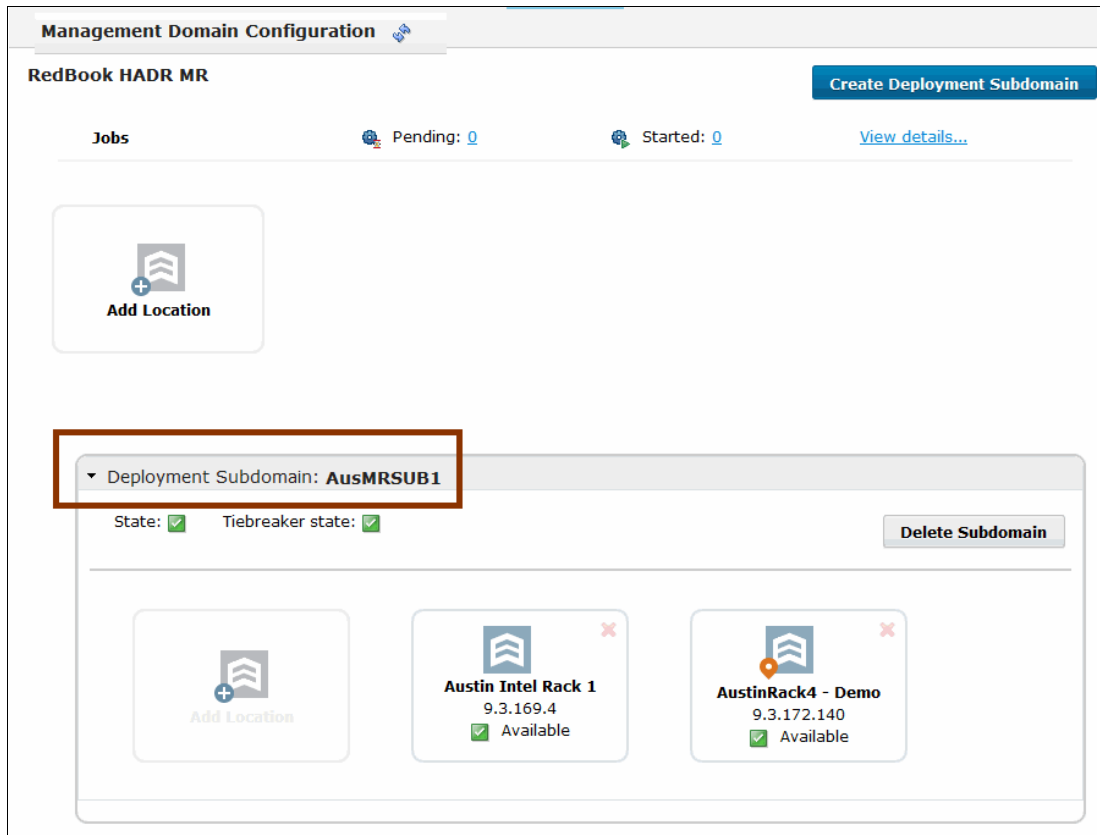


Figure 4-31 Deployment subdomain

Add systems (locations) to a deployment subdomain

Adding systems to the deployment subdomain is straightforward. After you add one or more locations (see “Add locations to the management domain” on page 75), you can then drag the system icons into the deployment subdomain as shown in Figure 4-32.

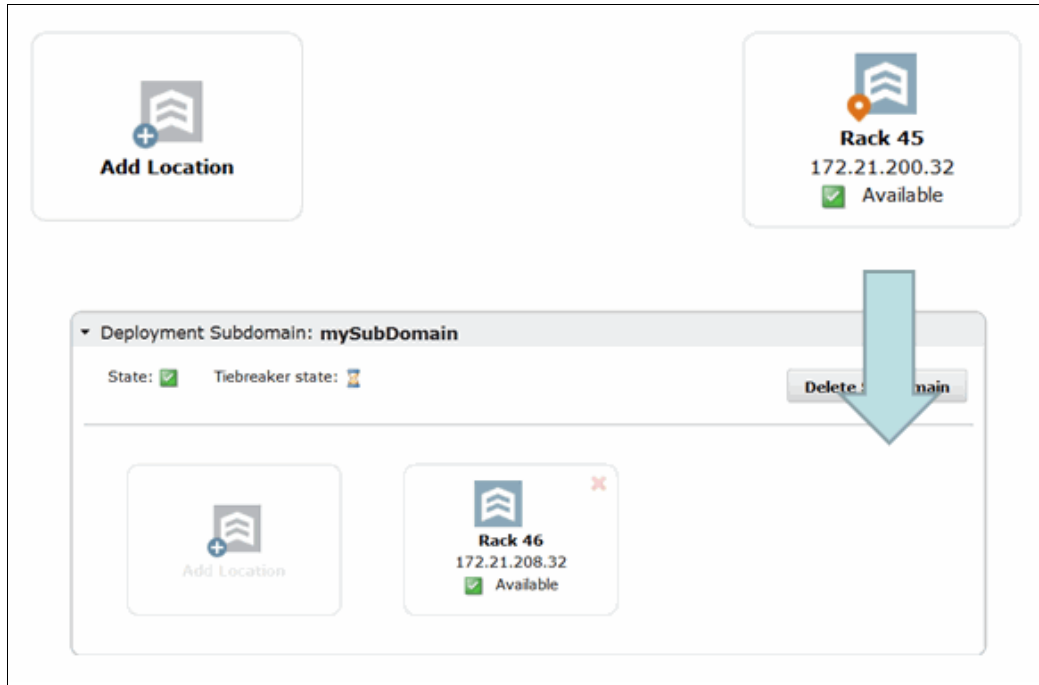


Figure 4-32 Add locations to deployment subdomain

Configure an iSCSI tiebreaker

An iSCSI tiebreaker is required to maintain data integrity and keep the deployment subdomain operational in a PureApplication System failure. It enables functionality in case members of a deployment subdomain are not able to communicate. The target must be at least 1 GB and exist on a server outside both racks. Figure 4-33 shows a sample iSCSI configuration.

Add Location to Deployment Subdomain

Deployment Subdomain Configuration

A separately managed iSCSI tiebreaker target (1 GB or greater) is required to maintain data integrity and help keep the subdomain functional in the event one or more PureApplication Systems becomes unreachable.

The iSCSI Qualified Name (IQN) of the PureApplication System might be required to configure the tiebreaker target.

The following are the iSCSI Qualified Names for the associated PureApplication Systems:

- iqn.1994-05.com.redhat:37a457c96232
- iqn.1994-05.com.redhat:14fcaaacd3bf
- iqn.1994-05.com.redhat:6e137dc07025
- iqn.1994-05.com.redhat:c4f6bc6c3178

Specify your iSCSI tiebreaker information below:

* iSCSI Tiebreaker IP Address:

* iSCSI Tiebreaker Qualified Name:

User:

Password:

Figure 4-33 iSCSI configuration

4.8.6 Cloud resources configuration

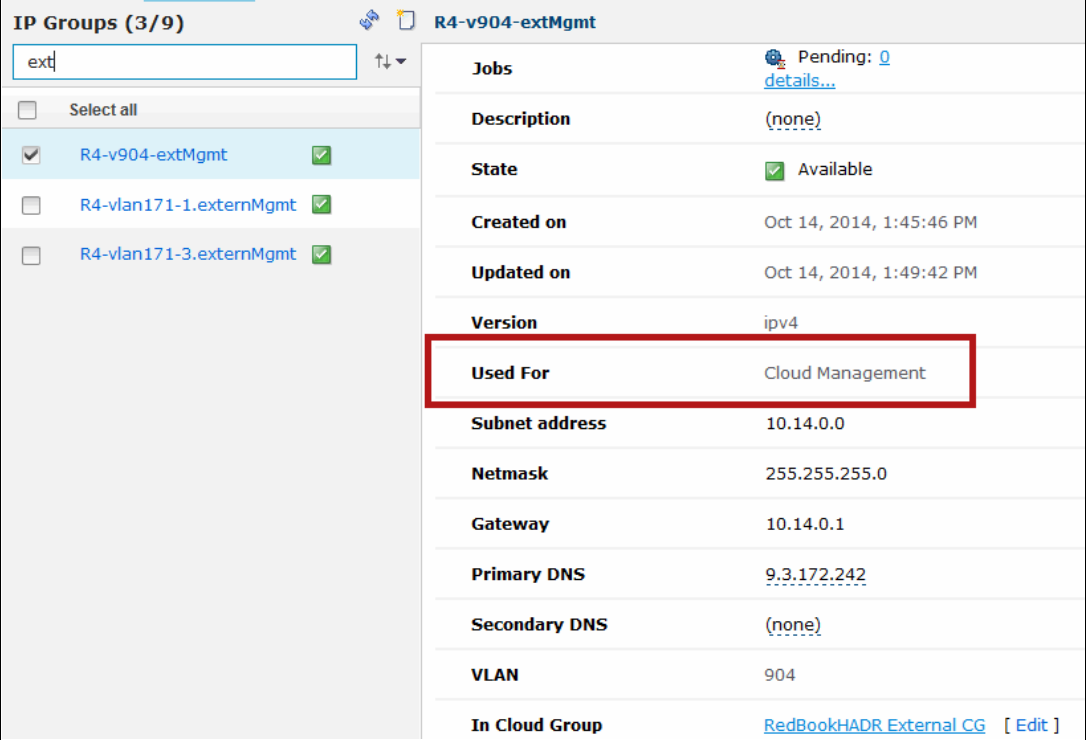
Multisystem environment configuration requires that cloud resources are configured to support management traffic across systems. For single system deployment, this traffic travels only on the internal IPv6. For multisystem environment deployment, this traffic travels between systems on the data center and requires external IP addresses.

Three cloud resources must be configured to use an external network for managing deployments across multiple system. Those cloud resources are listed below and discussed in detail in this section.

- ▶ IP groups
- ▶ Cloud groups
- ▶ Environment profiles

IP groups

You can configure an IP group to be used for either data communication or management data through the option **Used For**. For multisystem environment configurations, cloud management IP groups are required. See Figure 4-34 for an example of an IP group that was configured for this publication.



| IP Groups (3/9) | |
|---|--|
| <input type="checkbox"/> Select all | |
| <input checked="" type="checkbox"/> R4-v904-extMgmt <input checked="" type="checkbox"/> | |
| <input type="checkbox"/> R4-vlan171-1.externMgmt <input checked="" type="checkbox"/> | |
| <input type="checkbox"/> R4-vlan171-3.externMgmt <input checked="" type="checkbox"/> | |

| R4-v904-extMgmt | |
|-----------------------|--|
| Jobs | Pending: 0 details... |
| Description | (none) |
| State | <input checked="" type="checkbox"/> Available |
| Created on | Oct 14, 2014, 1:45:46 PM |
| Updated on | Oct 14, 2014, 1:49:42 PM |
| Version | ipv4 |
| Used For | Cloud Management |
| Subnet address | 10.14.0.0 |
| Netmask | 255.255.255.0 |
| Gateway | 10.14.0.1 |
| Primary DNS | 9.3.172.242 |
| Secondary DNS | (none) |
| VLAN | 904 |
| In Cloud Group | RedBookHADR External CG [Edit] |

Figure 4-34 IP group that is used for cloud management

In this example, the IP group named **R4-v904-extMgmt** is used for cloud management.

Cloud groups

For a cloud group to participate in a multisystem configuration, it must be configured so it can be managed by way of an external network. When you configure a cloud group, you are presented with two options for cloud management, either **By way of an internal network** or **By way of an external network**. Select **By way of an external network** so that cloud group management data travels on the data center network outside the system. An example of this configuration is shown in Figure 4-35. Notice that two IP groups that are used for cloud management are part of the cloud group.

The screenshot displays the configuration for a cloud group named "RedBookHADR External CG". The "Cloud Management" field is set to "By way of external network". Below this, a table lists associated IP groups, with two rows highlighted in red:

| Status | IP Groups | Used For | VLAN | Action |
|-----------|-------------------------|------------------|------|--------|
| Available | R4-vlan171-2 | Data | 171 | |
| Available | R4-v904-extMgmt | Cloud Management | 904 | |
| Available | R4-vlan171-3.externMgmt | Cloud Management | 171 | |
| Available | R4-v904-Data-10.14.1.x | Data | 904 | |

Figure 4-35 Cloud group that is managed by way of an external network

In this example, the cloud group named **RedBooksHADR External CG** performs its cloud management by way of an external network. Two associated IP groups are used for cloud management. One of the IP groups is shown in Figure 4-34 on page 79.

Environment profiles

In a manner similar to IP groups and cloud groups, you can configure environment profiles so that they are targets for virtual machines deployed to an externally managed network. The field that you must configure is named **Network** and can have values of either **Internally Managed** or **Externally Managed** as shown in Figure 4-36.

The screenshot shows the configuration page for the 'RedBook HADR External' environment profile. The 'Network' field is highlighted with a red box and set to 'Externally Managed'. Below, a table shows the 'Deploy to cloud groups' section with one entry: 'RedBookHADR External CG'.

| Name | Type | Location | Alias |
|-------------------------|--------|----------|----------------------------------|
| RedBookHADR External CG | Public | 1000202 | RedBookHADR External CG [remove] |

Figure 4-36 Externally managed environment profile

In this example, the externally managed environment profile named **RedBooks HADR External** deploys the cloud group named **RedBooks External CG**, which is a cloud group that is managed by way of an external network.

4.9 DNS setup for primary and secondary (cross) rack scenarios

In several scenarios, there are situations where applications (workloads) need to have the same host name on the primary data center (PDC) and secondary data center (SDC). This situation is needed to help recover possible unfinished in-flight transaction stored in transaction logs in replicated block storage. These duplicated network host names might cause host name conflict. In the case that the PDC is presumed unavailable, the original host names should not be used at the time the SDC uses them.

Host names need to be set aside and made exclusively accessible only for use by certain workloads on both PDC and SDC. In summary, IP groups must be allocated to own these host names on both sites. Additionally, a private DNS must be designated just for that special IP group at the SDC.

4.9.1 Network setup

The DNS configuration task must be done on both PDC and SDC. Complete these planning steps to implement the necessary DNS setup:

1. The deployer needs to decide how many IPs need to be reserved for the deployment.
2. The networking department must set aside IPs and host names for use at both the PDC and the SDC.
3. Create IP groups and environment profiles on both sites and restrict other users from using these resources.

After the planning work is done, the implementation work can be done in the following sequence:

1. Set up the DNS at the PDC.
2. Set up the IP group and environment profile at the PDC.
3. Set up the DNS at the SDC.
4. Set up the IP group and environment profile at the SDC.

The details of this configuration are shown in the following sections.

- ▶ Set up the DNS at the PDC
- ▶ Set up the IP group and environment profile at the PDC
- ▶ Set up the DNS at the SDC (includes Set up the IP group and environment profile at the SDC)

Set up the DNS at the PDC

There is no special DNS configuration that is needed at the PDC. The networking administrator needs to ensure all the reserved IP addresses are available and their host names are registered to the DNS.

Set up the IP group and environment profile at the PDC

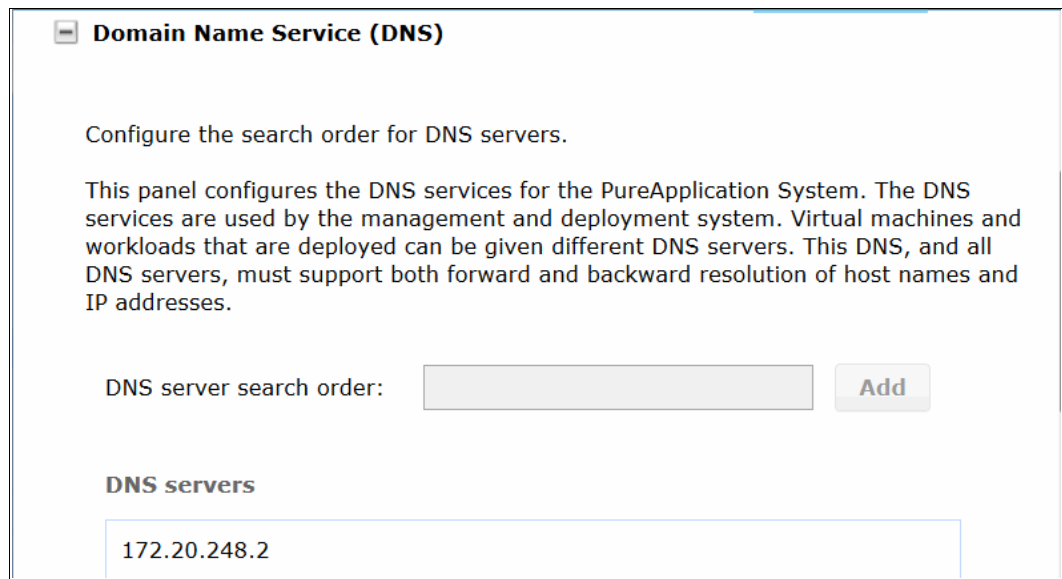
Complete these steps to set up the IP group and environment profile at the PDC:

1. Set up an IP group for the workload's DR operation.
2. Create an environment profile that contains the IP group, or add the IP group to an existing environment profile. This is done so that when deploying the patterns, the system can choose IP addresses automatically from the designated IP group.

Set up the DNS at the SDC

More care is needed when setting up the DNS at the SDC:

1. Create a DNS that maps all the reserved host names to the local IP addresses.
2. Configure PureApplication System to use that DNS. Details are shown in Figure 4-37.
 - a. From the System Console, select **System** → **Settings** → **Domain Name Service (DNS)**.
 - b. Expand that section and add the special DNS as the only DNS for the system.



Domain Name Service (DNS)

Configure the search order for DNS servers.

This panel configures the DNS services for the PureApplication System. The DNS services are used by the management and deployment system. Virtual machines and workloads that are deployed can be given different DNS servers. This DNS, and all DNS servers, must support both forward and backward resolution of host names and IP addresses.

DNS server search order:

DNS servers

- 172.20.248.2

Figure 4-37 Special DNS entry

3. Create an IP group as shown in Figure 4-38 and ensure that you have at least one IP address selected from the reserved IP range in it. You need to use the special DNS as the primary DNS and provide no secondary DNS.
 - a. From the System Console, select **Cloud** → **IP Groups**.
 - b. Click **New** in the upper left corner.
 - c. Enter the special DNS IP for the **Primary DNS**.
 - d. Add one IP address from the reserved list.
 - e. Fill the remainder of the required information.

Note: The requirement for adding at least one IP address is a temporary requirement only. A future release or fix pack will remove this requirement.

| IPAS HADR | | | | | | | | | | | |
|--|---|---|------------------------------------|--------|------------|-----------|--------|--|--------------|---|------------------------------------|
| Created on | Sep 24, 2014, 9:58:16 AM | | | | | | | | | | |
| Updated on | Oct 22, 2014, 11:51:28 PM | | | | | | | | | | |
| Version | ipv4 | | | | | | | | | | |
| Used For | Data | | | | | | | | | | |
| Subnet address | 172.20.64.0 | | | | | | | | | | |
| Netmask | 255.255.248.0 | | | | | | | | | | |
| Gateway | 172.20.64.1 | | | | | | | | | | |
| Primary DNS | 172.20.248.2 | | | | | | | | | | |
| Secondary DNS | (none) | | | | | | | | | | |
| VLAN | 1826 | | | | | | | | | | |
| In Cloud Group | Shared [Edit] | | | | | | | | | | |
| <div style="display: flex; align-items: center;"> ☰ IP addresses </div> <div style="display: flex; align-items: center; margin-top: 5px;"> <div style="display: flex; gap: 10px; margin-right: 20px;"> + ✖ 📄 🔗 </div> <div style="margin-right: 20px;"> <input type="text" value="Select a status"/> </div> <div> <input type="text" value="IP address"/> </div> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Status</th> <th style="width: 40%;">IP address</th> <th style="width: 40%;">Host name</th> <th style="width: 10%;">Action</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"></td> <td>172.20.71.40</td> <td>ipas-hadr-040.purescale.raleigh.ibm.com</td> <td style="text-align: center;">✖</td> </tr> </tbody> </table> | | | | Status | IP address | Host name | Action | | 172.20.71.40 | ipas-hadr-040.purescale.raleigh.ibm.com | ✖ |
| Status | IP address | Host name | Action | | | | | | | | |
| | 172.20.71.40 | ipas-hadr-040.purescale.raleigh.ibm.com | ✖ | | | | | | | | |

Figure 4-38 IP group that uses special DNS

4. Create an Environment Profile for deployments. Because the deployer at the SDC needs to pick a particular host name for each VM, do not set the **IP addresses provided by** field to **IP Group**. Instead, set the field to **Pattern Deployer**. The environment profile must include the special IP group designated for the deployment. See Figure 4-39 for details.
 - a. From Workload Console, select **Cloud** → **Environment Profiles**.
 - b. Be sure to select the IP group in the **Deploy to cloud groups** section.
 - c. Select *Pattern Deployer* for the **IP addresses provided by** field.

RB DR Only
Refresh Clone De

IP addresses provided by: Pattern Deployer ▼

Deployment priority: Platinum - High(16) Medium(8) Low(4) ▼

Time zone: Default time zone configured for the virtual image ▼

Language: Default language configured for the virtual image ▼

Deploy to cloud groups:

| Name | Type | Alias | |
|-------------------------------------|--|-----------|----------------|
| <input type="checkbox"/> Shared | Private | Shared | [remove] |
| <input type="checkbox"/> In use | Name | Alias | Subnet address |
| <input type="checkbox"/> | Share Data | Share | 172.20.64.0 |
| <input checked="" type="checkbox"/> | IPAS HADR Data | IPAS HADR | 172.20.64.0 |

Figure 4-39 Environment profile

4.10 Network configuration for WebSphere Application Server and DB2 scenarios

Because of specific DNS configurations need for several scenarios involving WebSphere Application Server and DB2, you must configure networking in the PDC and SDC differently depending on the scenario.

4.10.1 PDC

On the PDC side, set aside an IP group for WebSphere Application Server VMs. The name resolution of this IP group goes through the same site DNS chain setup for the PDC. There is no need for a special DNS for this IP group. This IP group is set up to reserve a set of host names for WebSphere Application Server VMs. Provide this set of host names to the SDC networking people so that they can configure an IP group designated for WebSphere Application Server VMs using those host names. Figure 4-40 shows an example of such an IP group. The IP group and IP HADR are set aside for WebSphere Application Server deployments only.

For this publication, IP addresses were set aside ranging from 172.20.95.1 to 172.20.95.40 and their corresponding host names from ipas-hadr-001.purescale.raleigh.ibm.com to ipas-hadr-040.purescale.raleigh.ibm.com for use by WebSphere Application Server VMs.










| IPAS HADR  Dele | | | |
|--|---|---|---|
| Netmask | 255.255.248.0 | | |
| Gateway | 172.20.88.1 | | |
| Primary DNS | 172.15.248.101 | | |
| Secondary DNS | 172.15.248.102 | | |
| VLAN | 833 | | |
| In Cloud Group | RBHADR_Internal_CG [Edit] | | |
| IP addresses | | | |
| <div style="display: flex; align-items: center; gap: 10px;">     </div> <div style="display: flex; align-items: center; gap: 10px;"> <div style="border: 1px solid #ccc; padding: 2px;">Select a status ▼</div> <div style="border: 1px solid #ccc; padding: 2px;">IP address</div> </div> | | | |
| Statu | IP address ▲ | Host name | Actions |
|  | 172.20.95.1 | ipas-hadr-001.purescale.raleigh.ibm.com |  |
|  | 172.20.95.2 | ipas-hadr-002.purescale.raleigh.ibm.com |  |

Figure 4-40 IP address range and corresponding host names

For the Environment Profile used by WebSphere Application Server, ensure that it includes the IP Group that was reserved. In this scenario, the IP Group is **IPAS® HADR**. Figure 4-41 shows how to include it in the environment profile.

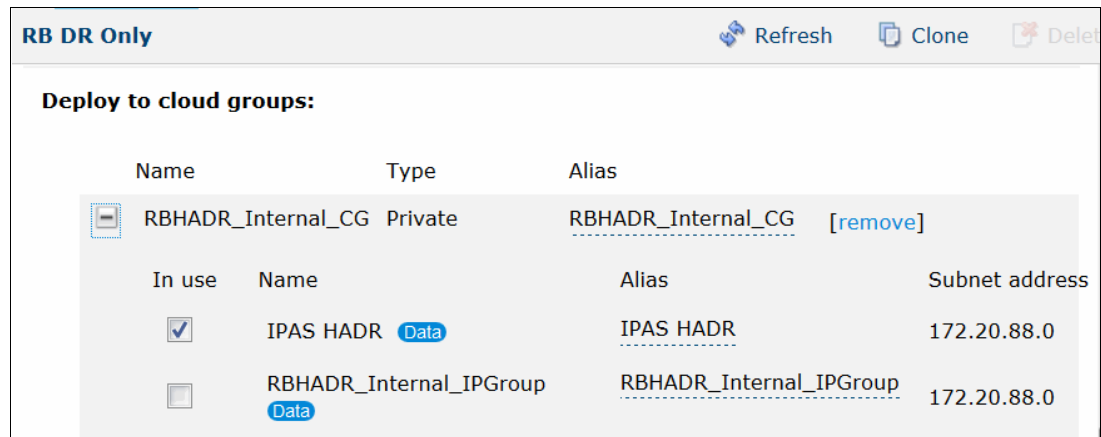


Figure 4-41 Include IP groups in environment profile

4.10.2 SDC

On the SDC side, the DNS setup is more complex than that on the PDC. The IP addresses assigned to WebSphere Application Server VMs must be assigned manually instead of automatically by the IP group. The deployer must find the IP address that has the exact host name for its corresponding VM. For instance, the host name of the WebSphere Custom Node1 on SDC must be the same as that of the Custom Node1 on PDC. If the host name is `ipas-hadr-010.purescale.raleigh.ibm.com` on the PDC, the deployer must find the IP address, for example `172.20.71.10`, for the host name on the SDC.

Because of that restriction, the host name resolution for this particular IP group cannot go through the common site DNS. Instead, it needs to have its own local DNS. Notice that in Figure 4-42 on page 88, the primary DNS is not used by other IP groups and there is no secondary DNS lookup for this IP group. Also, note that Figure 4-42 on page 88 shows that there is one IP address listed in the IP group. That does not mean that there is only one IP address for WebSphere Application Server to use. As you assign an address to a VM, that address is added to the list. You need to know the range of all available IP addresses that are reserved for WebSphere Application Server before you can assign one to a WebSphere Application Server VM at deployment time. The list in the interface shows only IPs that are occupied (being assigned), and thus not available for subsequent assignment.

Note: When you create the special IP group without specifying any available IP, the system returns an error and does not allow you to continue. This condition will be fixed in a future release. As a workaround, leave one IP in the IP group.

IPAS HADR Delete

Subnet address [172.20.64.0](#)

Netmask [255.255.248.0](#)

Gateway [172.20.64.1](#)

Primary DNS [172.20.248.2](#)

Secondary DNS [\(none\)](#)

VLAN 1826

In Cloud Group [Shared](#) [[Edit](#)]

IP addresses

Select a status
IP address

| Statu | IP address | Host name | Actions |
|-------|--------------|---|---------|
| | 172.20.71.40 | ipas-hadr-040.purescale.raleigh.ibm.com | |

Figure 4-42 Using local DNS and no secondary DNS

High availability and disaster recovery scenarios for DB2

This chapter describes how to implement DB2 solutions on IBM PureApplication System that provides High Availability and Disaster Recovery. A number of scenarios, as previously described in Chapter 3, “High availability and disaster recovery scenarios” on page 29, are continued in this chapter. Other scenarios exist, but this chapter covers only the ones that are tested and further documented within this publication.

Table 5-1 provides an overview of the four scenarios. All scenarios provide high availability (HA) through the DB2 high availability disaster recovery (HADR) mechanism. However, not all scenarios involve deployment across both the Primary Data Center (PDC) and the Secondary Data Center (SDC). The deployment has implications for disaster recovery (DR), which is only in place when DB2 is deployed across both data centers.

Table 5-1 Overview of the DB2 scenarios.

| Scenario | PDC | SDC | HA | DR |
|----------|-----|-----|-----|-----|
| DB2_1 | Yes | No | Yes | No |
| DB2_2 | Yes | Yes | Yes | Yes |
| DB2_3 | Yes | Yes | Yes | Yes |

This chapter assumes that the following listed items are in place:

- ▶ Two Intel W1500/W2500 IBM PureApplication Systems set up in different data centers
- ▶ Firmware 2.0.0.0 installed on both systems
- ▶ Block Storage Replication enabled as described in Chapter 4, “Infrastructure setup” on page 47
- ▶ DB2 10.5 Virtual System Patterns installed

Throughout this chapter, the *Default DB2 OLTP Pattern with HADR for Linux* shown in Figure 5-1, is used as a basis.

| Name | Version | Status | Created by | Updated by | Created on | Updated on | Actions |
|--|---------|--------|------------|------------|--------------------------|--------------------------|---------|
| Default DB2 BLU Datamart Pattern for Linux | 1.2.0.0 | Draft | admin | admin | Sep 17, 2014, 4:57:16 AM | Sep 17, 2014, 4:57:16 AM | [Icons] |
| Default DB2 Datamart Pattern for Linux | 1.2.0.0 | Draft | admin | admin | Sep 17, 2014, 4:57:09 AM | Sep 17, 2014, 4:57:09 AM | [Icons] |
| Default DB2 OLTP Pattern for Linux | 1.2.0.0 | Draft | admin | admin | Sep 17, 2014, 4:56:50 AM | Sep 17, 2014, 4:56:50 AM | [Icons] |
| Default DB2 OLTP Pattern with HADR for Linux | 1.2.0.0 | Draft | admin | admin | Sep 17, 2014, 4:57:02 AM | Sep 17, 2014, 4:57:02 AM | [Icons] |

Figure 5-1 DB2 Virtual System Patterns available on W1500/W2500 PureApplication System 2.0.0.0

Unlike the *classic* Virtual System Patterns, this pattern no longer uses the Hypervisor Edition images. Instead, a Software Component is used to install the DB2 product on an image that contains only the operating system. As described in 1.2.6, “Multisystem deployments” on page 8, this approach brings several advantages including *multi-target deployment*, whereby a single pattern can be deployed across two PureApplication Systems.

Note: The instructions provided here are specifically for the Intel W1500/W2500 models. However, these scenarios should work equally on the Power W1700/W2700 models, although there can be minor differences in the instructions.

This chapter includes the following sections:

- ▶ Introduction to DB2 HADR
- ▶ DB2 Client Setup
- ▶ Building a DB2 Virtual System Pattern
- ▶ Scenario DB2_1
- ▶ Scenario DB2_2: Two systems using a single pattern
- ▶ Scenario DB2_3: Two systems using block storage replication
- ▶ Validation
- ▶ Testing for outages

5.1 Introduction to DB2 HADR

DB2 HADR is a feature available in DB2 10.5 Enterprise Edition and earlier releases. This mechanism allows a single DB2 database to be deployed across two hosts and connected over the network. Although this is not something within the scope of this book, some of the fundamentals are covered briefly. The following link provides more details about DB2 HADR:

http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/c0011724.html

5.1.1 Primary, standby, and log shipping

When configured for HADR, a DB2 database on each system gets assigned the role of *primary* or *standby*. DB2 clients connect to the primary database, but transactions committed there are sent across to the standby database by shipping the transactions logs. This process is also referred to as *log shipping*. DB2 HADR provides the following four listed options to perform the log shipping:

- ▶ Super asynchronous
- ▶ Asynchronous
- ▶ Near synchronous
- ▶ Synchronous

Figure 5-2 shows the four options for log shipping.

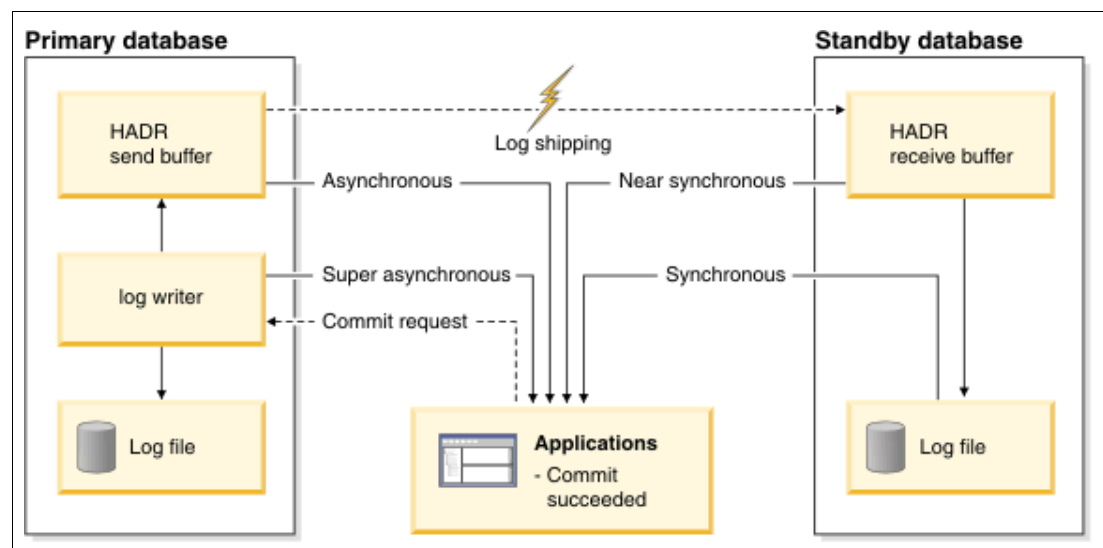


Figure 5-2 DB2 HADR provides four different options to keep the standby in sync with the primary

Each option has different characteristics. Synchronous ensures absolute consistency between the primary and the standby, however, this comes at a cost of increased latency. The other options cannot ensure consistency, but have a smaller increase in latency.

5.1.2 DB2 client and automatic client rerouting

A DB2 database that is configured for HADR requires the DB2 client to connect to the primary. However, if the primary has a planned or unplanned outage, the standby takes over and effectively becomes the new primary.

A DB2 client normally automatically retrieves the host name and port of the standby server when it connects to the DB2 database of the primary for the first time. This information ensures that the DB2 client can automatically fail over its connection to the standby server in case of an outage. This mechanism is also known as DB2 *Automatic Client Rerouting* (ACR).

In a scenario where the DB2 client has never connected to the primary before an outage occurs, ACR does not allow the DB2 client to connect to the new primary (standby). Depending on the type of DB2 client, there are ways to prevent this occurrence. For example, when using the DB2 JCC JDBC Provider, you can specify the host name and port for the standby server as custom properties on the data source. You can find more details about that process by using the following link:

http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.apdv.java.doc/src/tpc/imjcc_c0056186.html

ACR is a complex subject. Additional information and more background is available by using the following links:

<http://www.ibm.com/support/docview.wss?uid=swg21394840>

<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/DB2HADR/page/Client%20Reroute>

5.2 DB2 Client Setup

To be able to perform failover scenarios with DB2, a DB2 client is needed to prove the availability of a DB2 database. For example scenarios where WebSphere and DB2 are combined (which means WebSphere is the DB2 client), see 3.4, “HADR scenarios for WebSphere Application Server and DB2” on page 39.

This chapter uses a simple DB2 CLI client. This following section describes how this client is set up and configured.

5.2.1 Deploy DB2 client on PureApplication System

To quickly set up a DB2 client, a simple DB2 Virtual System Pattern is built and deployed. This pattern can be used to perform a DB2 server installation and set up a DB2 instance, and includes a DB2 client. The scenario that is used in this chapter does not use the DB2 server or instance, but does use the DB2 client run time.

Note: Using the same DB2 Software Component for the DB2 client has an advantage. The client is using the exact same version of DB2 as the DB2 HADR server environments that are used for the various scenarios.

Figure 5-3 shows the scenario's DB2 client Virtual System Pattern.

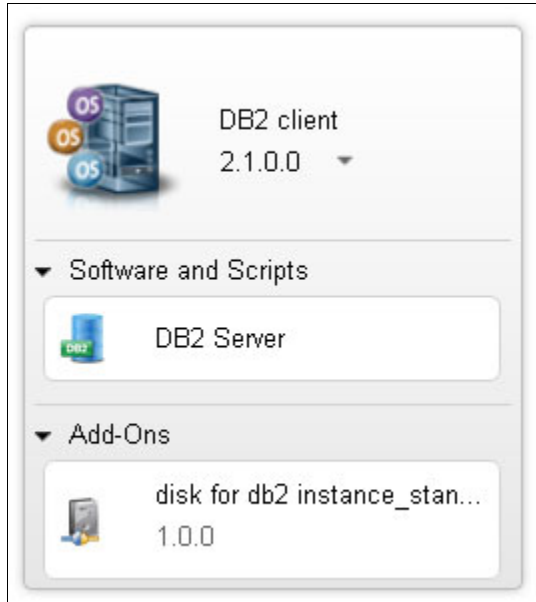


Figure 5-3 DB2 Virtual System Pattern to deploy a DB2 client

5.2.2 Configure DB2 client

Before connecting to a DB2 database, you must configure the DB2 client. See 5.7.1, “Configure the DB2 client” on page 160 for more details about how to do this configuration.

5.3 Building a DB2 Virtual System Pattern

As mentioned at the start of this chapter, the *Default DB2 OLTP Pattern with HADR for Linux* is used for the DB2 scenarios that are described in this chapter. However, a number of small modifications to this Virtual System Pattern are needed. Those changes are described in this section.

5.3.1 Cloning the Default DB2 OLTP Pattern with HADR for Linux

Instead of making the changes directly in the original pattern provided by IBM, this process creates a copy of the pattern and uses the following steps to make the changes in the copy:

1. From the Workload Console, select **Patterns** → **Virtual System Patterns** and filter for Default DB2 (see Figure 5-4).

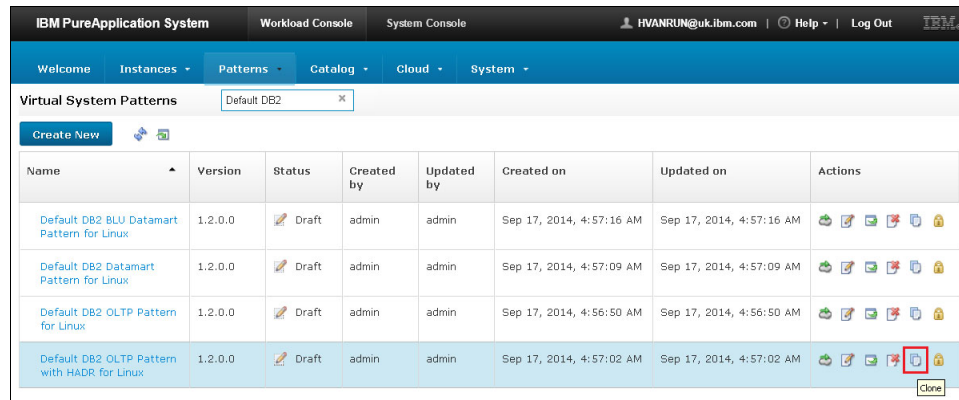


Figure 5-4 Filter for Default DB2

2. Click the **Clone** icon (shown in Figure 5-4 in the lower right) and specify a name for the new pattern. This example uses **ITSO DB2 OLTP HADR Pattern** for the name. Also, ensure that **Core OS 2.1.0.0 42** has been selected as the virtual image (see Figure 5-5).

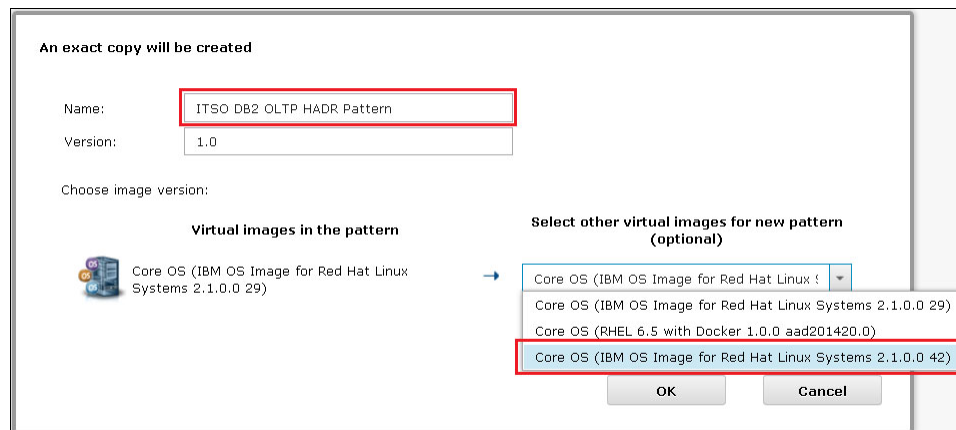


Figure 5-5 Configuring the clone

3. After a short time, you should see a notification that the new pattern has been created (see Figure 5-6).

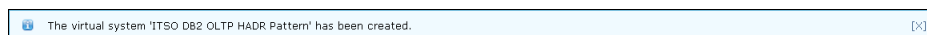


Figure 5-6 Notification message

- Go to **Patterns** → **Virtual System Patterns**, and this time filter for ITS0. You should see the newly created pattern appear (see Figure 5-7).

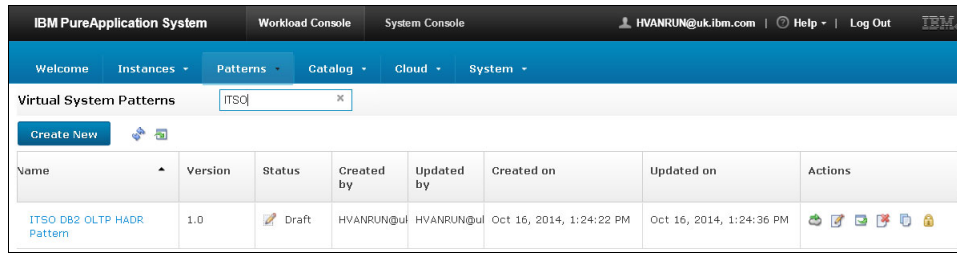


Figure 5-7 New pattern verification

5.3.2 Modifying the new pattern, ITS0 DB2 OLTP HADR Pattern

With the new pattern created, you can start modifying it. The changes described in this section are for the test purposes of this book. For actual implementations, more modifications might be required including the use of block storage.

Note: The use of *block storage* with DB2 on PureApplication System 2.0 can bring a number of tangible benefits, including the ability to clone and resize the actual storage. The use of block storage also allows for replication of data to another system.

Using block storage is simple. You can replace the **Default add disk** add-on with **Default attach block disk**.

To modify the pattern, complete the following steps:

- From the Workload Console, select **Patterns** → **Virtual System Patterns**. Find the newly created pattern by filtering for ITS0 and click the **Edit** icon to open the pattern in the Pattern Builder (see Figure 5-8).

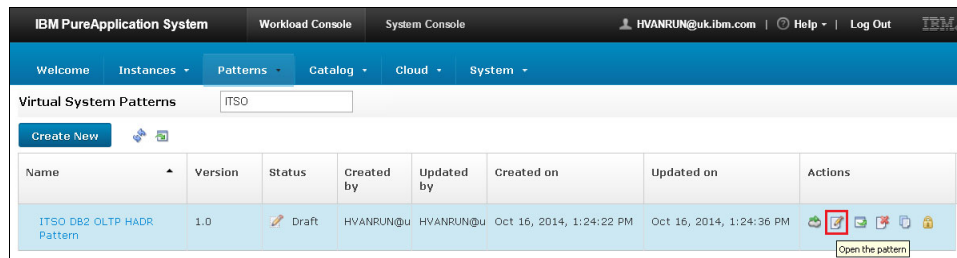


Figure 5-8 Edit the pattern

2. The Pattern Builder opens in a new window, you should recognize the name of the pattern that is shown in the right corner (see Figure 5-9).

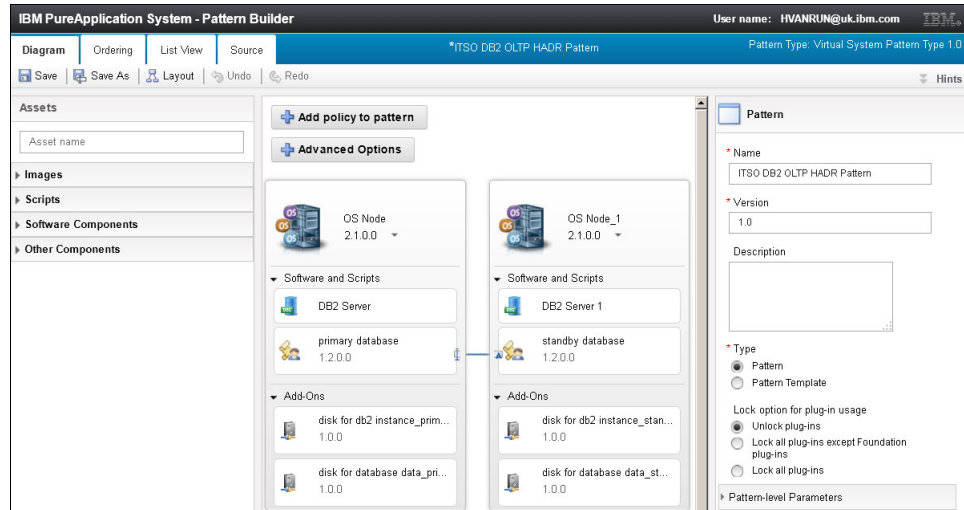


Figure 5-9 Pattern in the Pattern Builder

3. Next, rename the parts (Virtual Machines), the Software Components, and the Script Packages as noted in the following actions: (see Figure 5-10 on page 97)
 - a. Rename part OS Node to DB2 VM one.
 - b. Rename part OS Node 1 to DB2 VM two.
 - c. Rename Software Component DB2 Server to DB2 Server one.
 - d. Rename Software Component DB2 Server 1 to DB2 Server two.
 - e. Rename Script Package primary database to primary database - ITS0.
 - f. Rename Script Package standby database to standby database - ITS0.

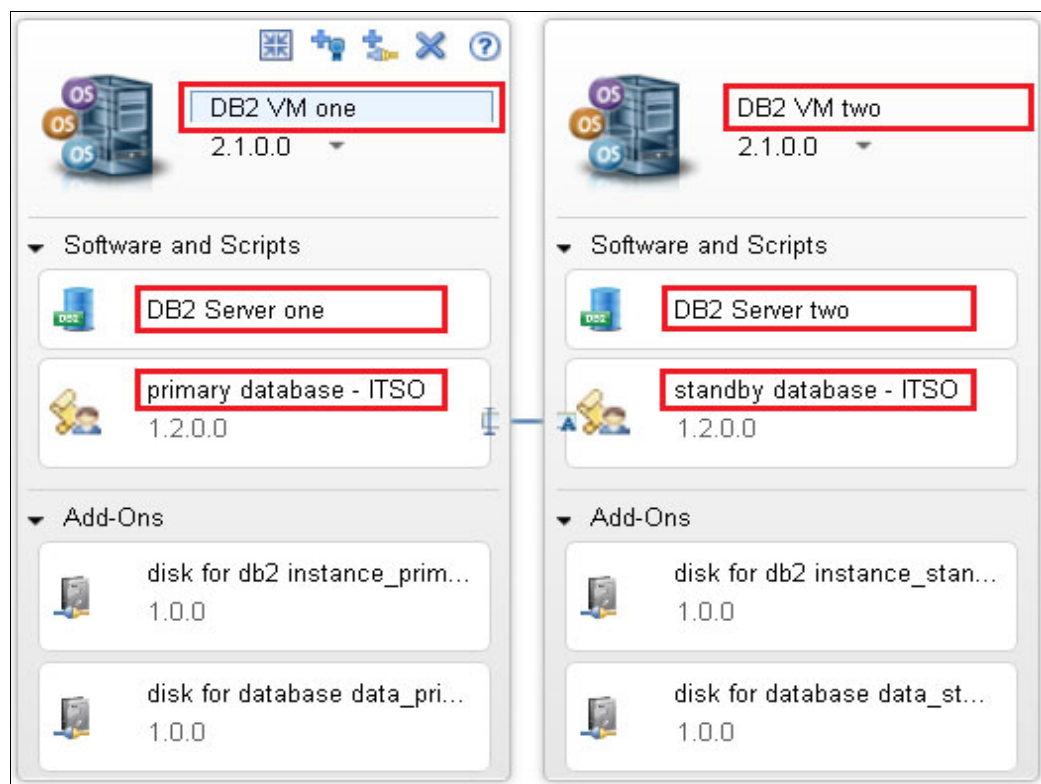


Figure 5-10 Renaming parts, software components, and script packages

- Now review, modify, and lock the other pattern-level parameters. Pattern level parameters are defined at the pattern level and can be referenced from Script Packages and Add-ons. Table 5-2 shows the pattern parameters, values, and whether they are locked or not.

Table 5-2 Overview of pattern parameters and locking information

| Parameter | Value | Locked? |
|-------------------------|----------|---------|
| databaseName | itso | Yes |
| databaseUser | itsouser | Yes |
| Password (root) | | No |
| Password (virtuser) | | No |
| Virtual CPUs | 1 | Yes |
| Memory size (MB) | 4096 | Yes |
| Reserve physical CPUs | False | Yes |
| Reserve physical memory | True | Yes |
| Fenced user group | db2fadm1 | Yes |
| Fenced user | db2fenc1 | Yes |
| Password (Fenced user) | | No |
| Instance owner group | db2iadm1 | Yes |
| Instance name | db2inst1 | Yes |

| Parameter | Value | Locked? |
|-----------------------------|---------------|---------|
| Password (Instance owner) | | No |
| Database compatibility mode | DB2 (default) | Yes |
| DB2 Service Port | 50000 | Yes |
| localHADRSevPort | 55555 | Yes |
| remoteHADRSevPort | 55555 | Yes |
| databaseImage | | Yes |
| databaseUserPassword | | No |
| databaseTerritory | US | Yes |
| databaseCodeset | UTF-8 | Yes |
| databasePageSize | 4 | Yes |
| databaseCollate | SYSTEM | Yes |

5. Set all the pattern level parameters, as shown previously in Table 5-2 on page 97, by repeating the following steps used to set the parameter databaseName:
 - a. Find the parameter databasaName in the Pattern Builder (under Pattern-level parameters located to the right side of the window).
 - b. Change the value to its o.
 - c. Click the **Lock** icon to lock the value of this parameter. This helps speed up deployment as you will not have to specify or review this aspect at time of deployment (see Figure 5-11).

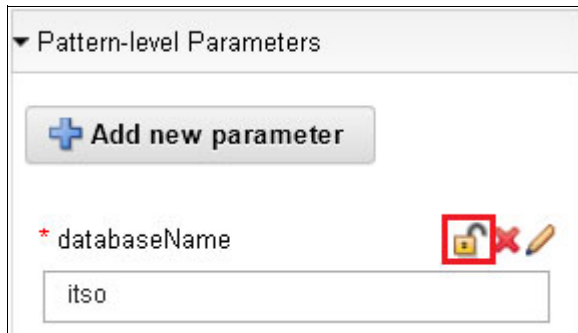


Figure 5-11 Locking a parameter value

6. Finally, to simplify deployment of the pattern, lock the parameters for the disk add-ons noted in the following list: (see Figure 5-12 on page 99)
 - disk for db2 instance_primary
 - disk for db2 instance_standby
 - disk for database data_primary
 - disk for database data_standby

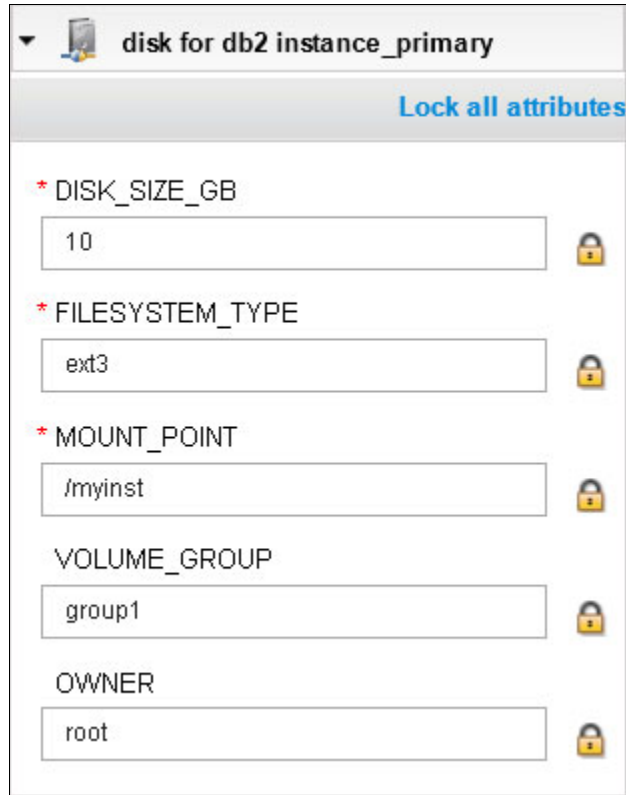


Figure 5-12 Lock disk add-ons to save time at deployment

7. With all the changes made, click **Save** to save the pattern

5.3.3 Optional: Multiple HADR databases

The pattern created in the previous section can easily be extended to deploy multiple HADR databases using the same pattern. This scenario is common for many clients. To complete this option, build a new pattern using the pattern (ITS0 DB2 OLTP HADR Pattern) by following these steps:

1. From the Workload Console, select **Patterns** → **Virtual System**. Find the pattern ITS0 DB2 OLTP HADR Pattern by filtering for ITS0 and click the **Clone** icon to create a copy. The Clone icon is in the lower right corner (see Figure 5-13).

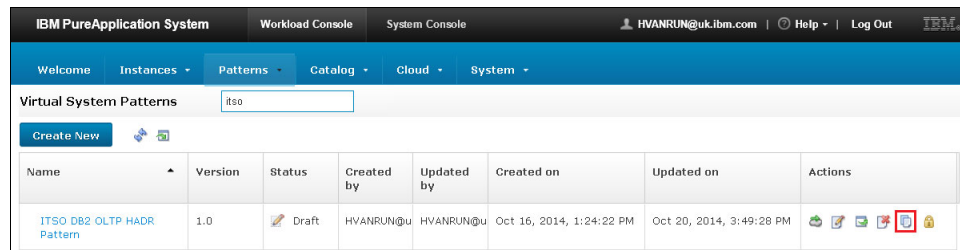


Figure 5-13 Filtering and cloning the pattern

2. Provide a name for the new pattern, for example **ITSO DB2 OLTP HADR Pattern with two databases**. Also, make sure that the image **IBM OS Image for Red Hat Linux Systems 2.1.0.0 42** is used (see Figure 5-14).

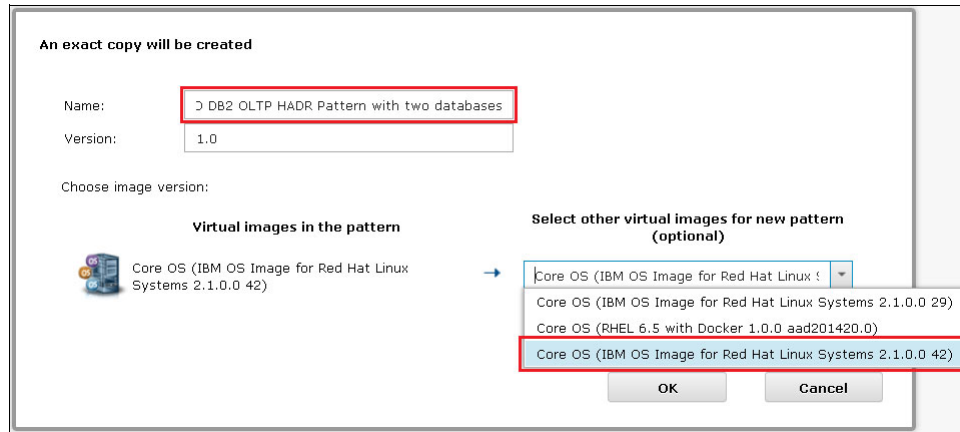


Figure 5-14 Naming pattern and selecting virtual image

3. After a short time, you should see a notification that the new pattern has been created (see Figure 5-15).

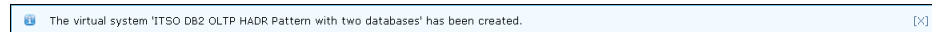


Figure 5-15 Verification of pattern creation

4. Select **Patterns** → **Virtual System Patterns** and filter for **ITSO**.
5. Select the newly created pattern, **ITSO DB2 OLTP HADR Pattern with two databases**.
6. Click the **Edit** icon to open the pattern in the Pattern Builder (see Figure 5-16).

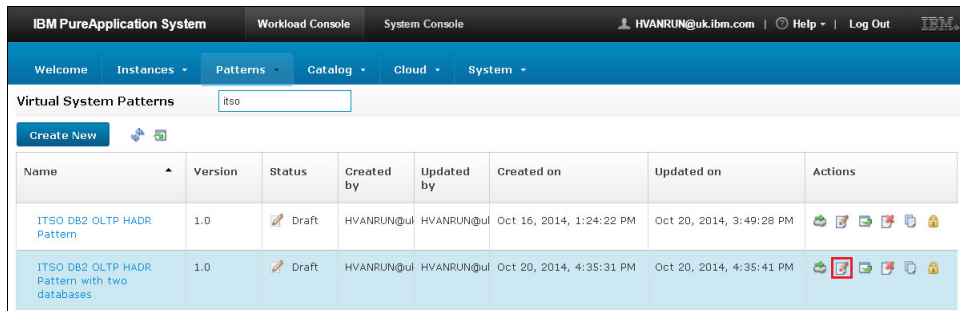


Figure 5-16 Editing the newly created pattern

7. With the pattern opened in the Pattern Builder, add two Script Packages to the pattern for the second HADR database by using the following steps:
 - a. Expand the Scripts section on the left side and filter for HADR. You then see the Script Packages, CreateDB_HADRPrimary and CreateDB_HADRStandby. These are the Script Packages that you are adding to the pattern (see Figure 5-17).

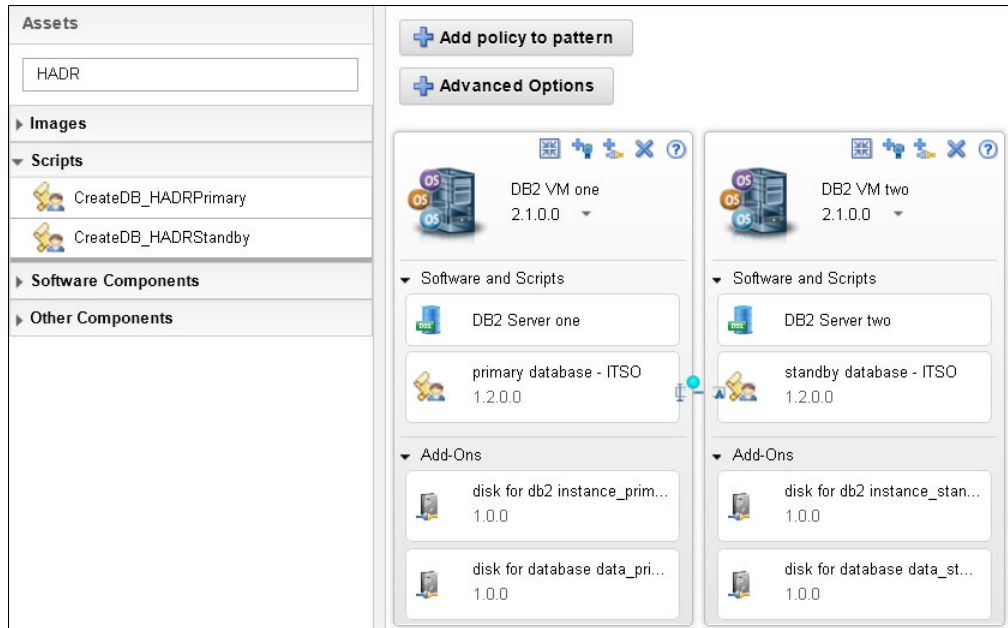


Figure 5-17 Identifying Script Packages by filtering for HADR

- b. Add the Script Package CreateDB_HADRStandby to DB2 VM two, by dragging the script package and dropping it onto that VM (see Figure 5-18).
- c. Add the Script Package CreateDB_HADRPrimary to DB2 VM one, by dragging the script package and dropping it onto that VM (see Figure 5-18).

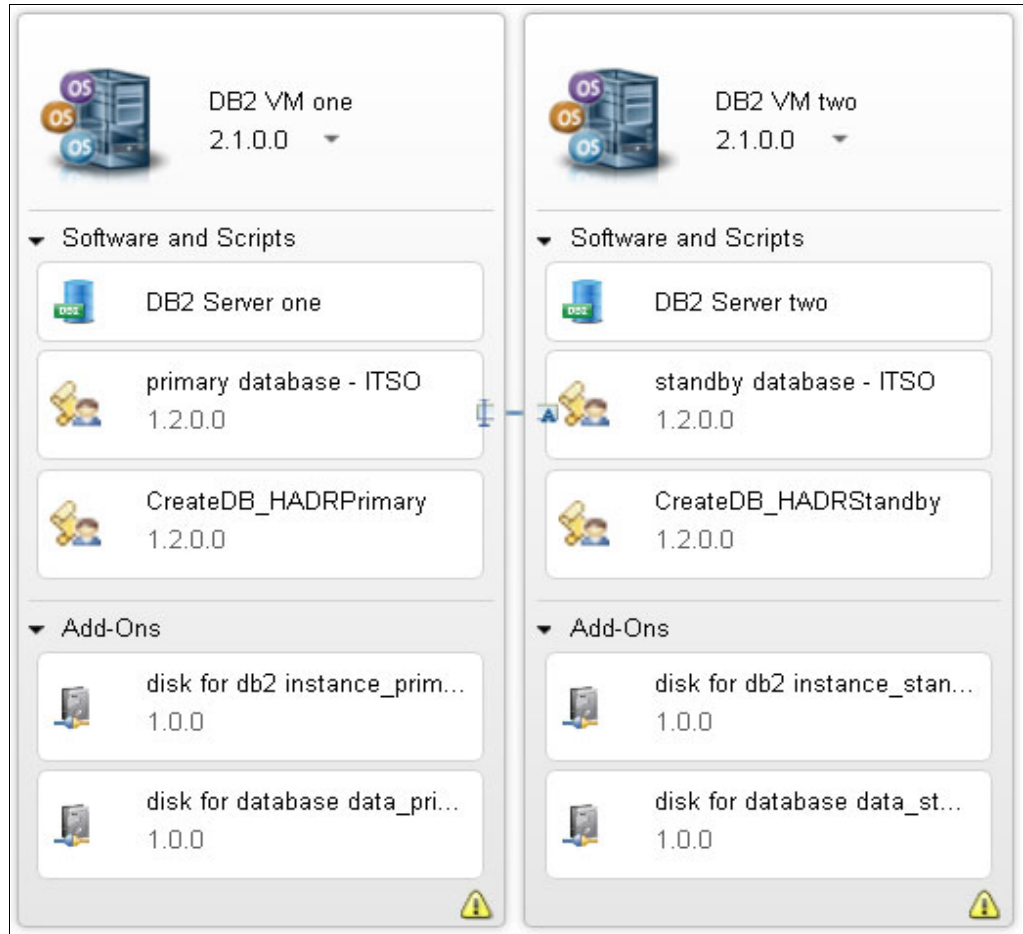


Figure 5-18 Adding script packages

8. Rename the Script Packages just added to reflect the name of the second HADR database: ITS02:
 - a. Rename CreateDB_HADRStandby, in DB2 VM two, to standby database - ITS02.
 - b. Rename CreateDB_HADRPrimary, in DB2 VM two, to primary database - ITS02.

9. With the script packages renamed, you now need to define and change some of the script packages parameters. Change the parameter, Database name, for the Script Package standby database - ITSO by using the following steps: (see Figure 5-19):
 - a. Find the parameter in the right side and click the **Delete** icon to delete the current value, `${pattern.databaseName}`. Note that this is a reference to a pattern-level parameter, so you are effectively breaking the reference.

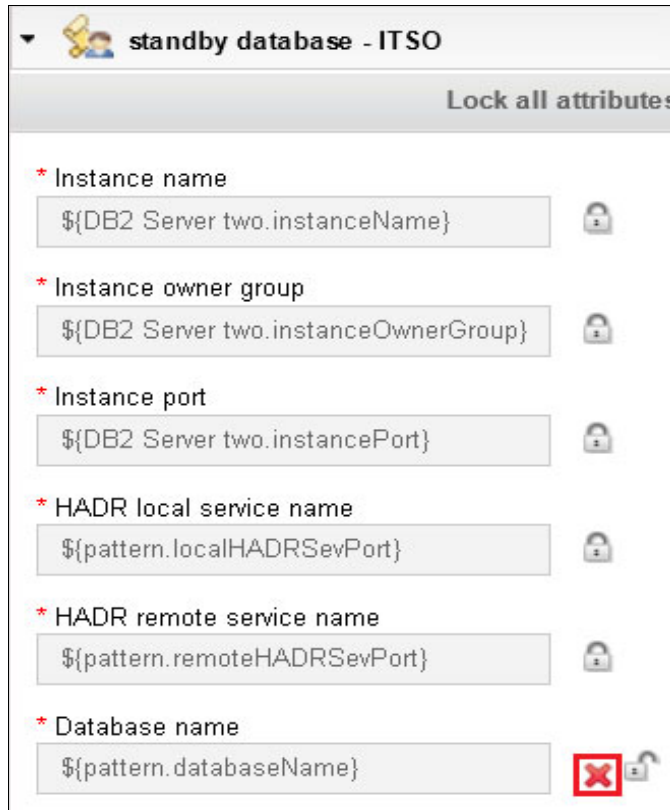


Figure 5-19 Changing the parameter for database name

- b. With the value removed, now specify `itso` as the name for the database.

Note: When renaming, do not use capital letters because the Script Package only accepts a string without any capitals. The actual name of the DB2 database is not case sensitive. Also, keep in mind that the name of a DB2 database is limited to eight characters.

- c. Finally, lock the parameter by clicking the **Lock** icon (see Figure 5-20).



Figure 5-20 Locking the database name parameter

10. In a similar fashion, make changes for the parameters of the Script Packages for the ITSO database, as shown in Table 5-3.

Table 5-3 Values for Script Package parameters

| | Standby database - ITSO | Primary database - ITSO |
|--------------------------|-------------------------|--|
| Database name | itso | \${standby database - ITSO.databaseName} |
| HADR local service name | 55555 | 55555 |
| HADR remote service name | 55555 | 55555 |

11. Now you need to populate the parameters for the Script Packages of the ITSO2 database. It is not uncommon to notice a *Warning* icon on both virtual machines in the Pattern Builder shown in Figure 5-21. This is a result of some parameters (properties) not being defined for the Script Packages of the ITSO2 database.

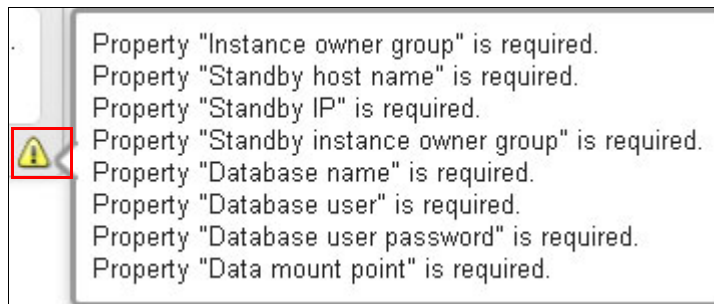


Figure 5-21 Warnings in the Pattern Builder highlighting parameters that need to be defined

12. To update the parameters of the ITSO2 database, the steps are similar to the ones previously described. However, they now include references to pattern-level parameters. Use the following steps to continue this process:

- a. Change the parameter, Instance name, for the Script Package standby database - ITSO2. To do so, find the parameter and click the **Add reference** icon (see Figure 5-22).

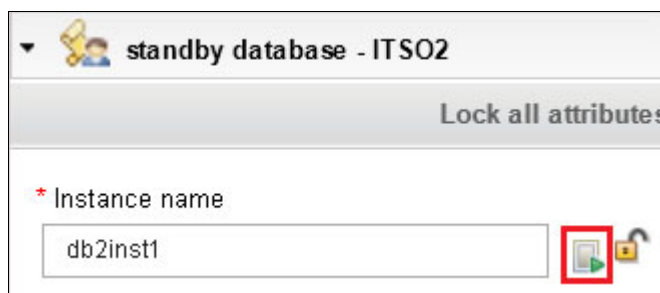


Figure 5-22 Changing the parameter for Instance name

- b. In the dialog window, select **component-level parameter**. Select **DB2 Server two** as the component, and **instanceName** as the **Output** attribute. Click **Add** to generate the Output value. Your results should be similar to those shown in Figure 5-23.

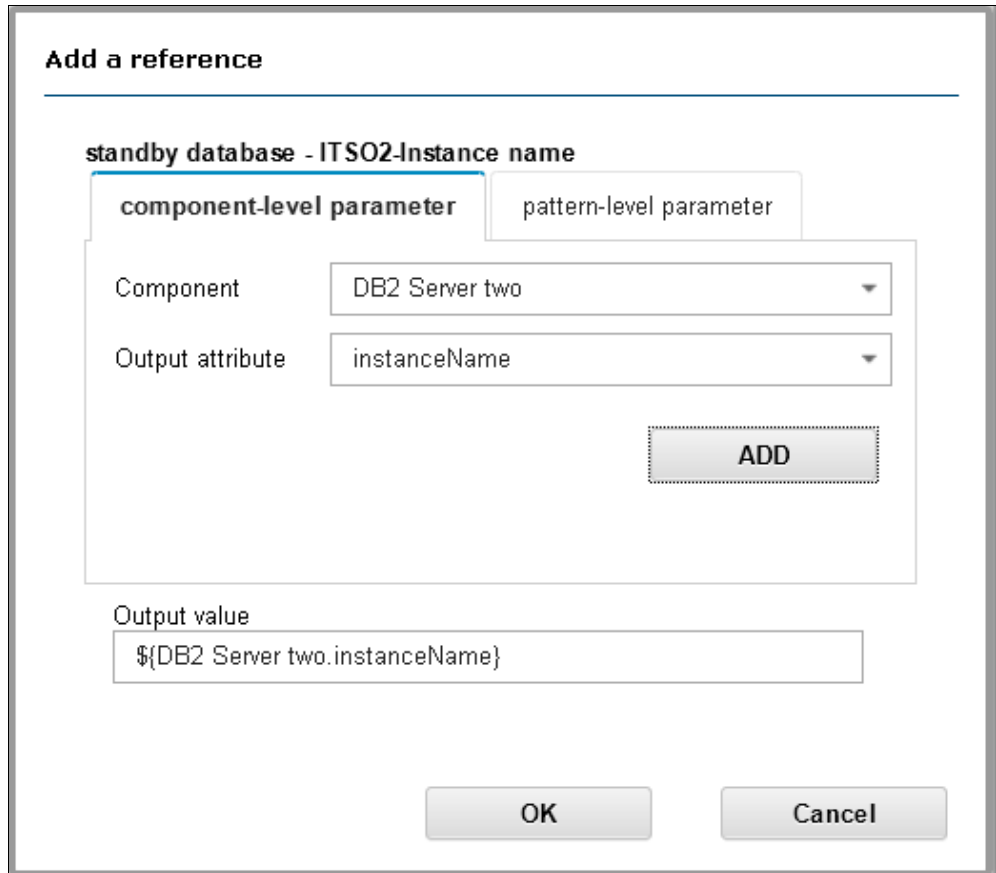


Figure 5-23 Adding a reference

- c. Click **OK** to add the reference. You should notice that the value of the parameter is now grayed out and locked, which is normal when you add references (see Figure 5-24).

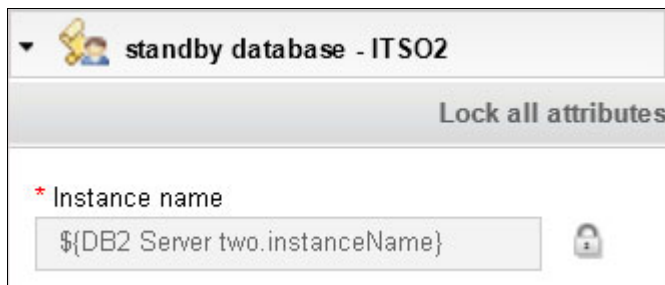


Figure 5-24 Parameter with value locked by adding a reference

13. Complete the list of parameters for the Script Packages, standby database - ITS02 and primary database - ITS02, as shown in Table 5-4. Some of these parameters are only applicable to the primary database - ITS02.

Table 5-4 Values and references for Script Package parameters.

| Parameter | Standby database - ITS02 | Locked? | Primary database - ITS02 | Locked? |
|--------------------------|---------------------------------------|---------|---|---------|
| Instance name | \${DB2 Server two.instanceName} | Yes | \${DB2 Server one.instanceName} | Yes |
| Instance owner group | \${DB2 Server two.instanceOwnerGroup} | Yes | \${DB2 Server one.instanceOwnerGroup} | Yes |
| Instance port | \${DB2 Server two.instancePort} | Yes | \${DB2 Server one.instancePort} | Yes |
| HADR local service name | 55556 | Yes | 55556 | Yes |
| HADR remote service name | 55556 | Yes | 55556 | Yes |
| Standby host name | (N/A) | (N/A) | \${standby database - ITS02.host} | Yes |
| Standby IP | (N/A) | (N/A) | \${standby database - ITS02.ipaddr} | Yes |
| Standby instance name | (N/A) | (N/A) | \${standby database - ITS02.instanceName} | Yes |
| Standby owner group | (N/A) | (N/A) | \${standby database - ITS02.instanceOwnerGroup} | Yes |
| Standby instance port | (N/A) | (N/A) | \${standby database - ITS02.instancePort} | Yes |
| HADR sync mode | (N/A) | (N/A) | SYNC | No |
| Database name | itso2 | Yes | itso2 | Yes |
| Description | | Yes | | Yes |
| Database image | \${pattern.databaseImage} | Yes | \${pattern.databaseImage} | Yes |
| Database user | \${pattern.databaseUser} | Yes | \${pattern.databaseUser} | Yes |

| Parameter | Standby database - ITSO2 | Locked? | Primary database - ITSO2 | Locked? |
|------------------------|--|---------|--|---------|
| Database user password | \${pattern.databaseUserPassword} | Yes | \${pattern.databaseUserPassword} | Yes |
| Data mount point | \${disk for database data_standby.MOUNT_POINT} | Yes | \${disk for database data_primary.MOUNT_POINT} | Yes |
| Database territory | \${pattern.databaseTerritory} | Yes | \${pattern.databaseTerritory} | Yes |
| Database codeset | \${pattern.databaseCodeset} | Yes | \${pattern.databaseCodeset} | Yes |
| Database page size | \${pattern.databasePageSize} | Yes | \${pattern.databasePageSize} | Yes |
| Database collate | \${pattern.databaseCollate} | Yes | \${pattern.databaseCollate} | Yes |

14. The following pattern-level parameters can now be deleted because they no longer serve a purpose. Also, note that they are not referenced by any of the Script Packages:

- databaseName
- localHADRSevPort
- remoteHADRSevPort

a. To delete these parameters, go the Pattern-level parameters, find the parameter, and click the **Delete** icon to remove them (see Figure 5-25).

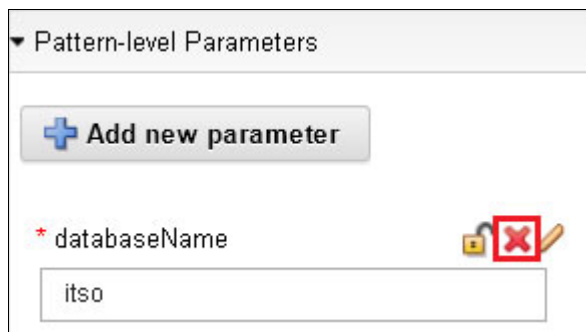


Figure 5-25 Deleting pattern-level parameters

- b. You should see that no mappings are present for the parameter. Confirm the deletion of the parameter by clicking **OK** (see Figure 5-26).

Pattern-level Parameter

Delete Pattern-level parameter

* Label:

* Parameter Type:

* ID:

Description:

Default Value:

No mapping(s) present
Do you want to delete the parameter?

Figure 5-26 Confirmation of no mapping and deletion of parameter

5.4 Scenario DB2_1

For this scenario, you deploy a DB2 primary and standby database to a single PureApplication System in the PDC. Figure 5-27 shows the scenario. DB2-P stands for the primary database, and DB2-S for the standby database.

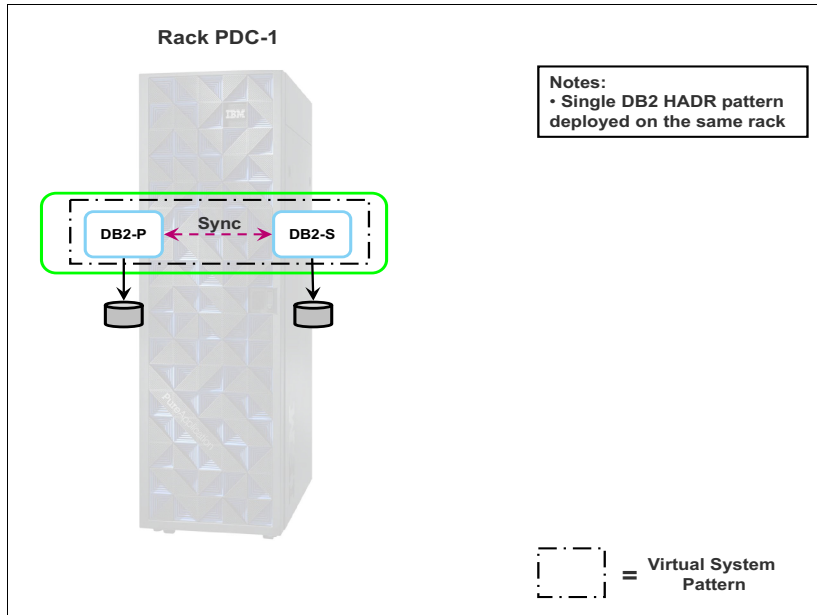


Figure 5-27 Scenario DB2_1 - DB2 primary and standby on the same system in the PDC

5.4.1 Deployment

For this scenario, you deploy the Virtual System Pattern, ITSO DB2 OLTP HADR Pattern. See 5.3, “Building a DB2 Virtual System Pattern” on page 93 for more details about this Virtual System Pattern. Figure 5-28 shows the ITSO DB2 OLTP HADR Pattern. Note that block storage is not used in this pattern.

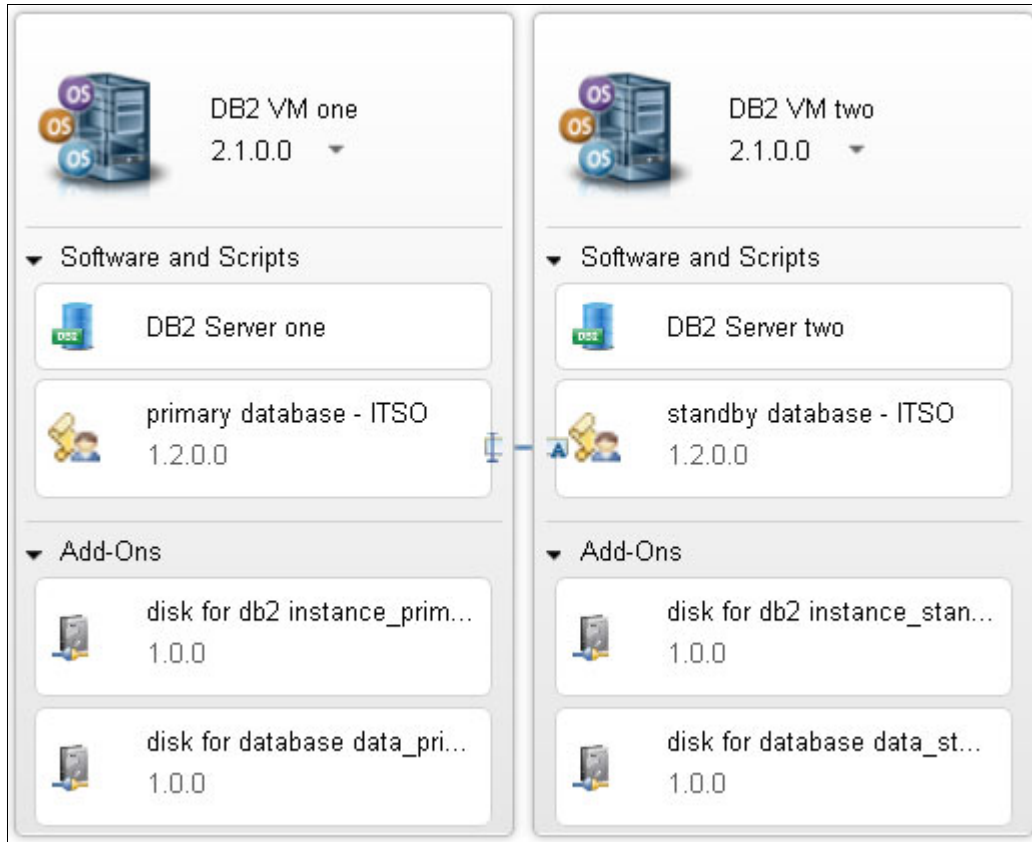


Figure 5-28 Virtual System Pattern, ITSO DB2 OLTP HADR Pattern

For this deployment, use the environment profile named *Redbooks HADR Internal Profile*. This is a standard environment profile, which does not allow for pattern deployments across multiple PureApplication Systems. This type of environment profile uses an *internally managed* network, as shown in Figure 5-39 on page 115. The status of the environment profile must also be valid. The status is valid if the current status shows as: Environment profile can now be used for deployments, also shown in Figure 5-29.

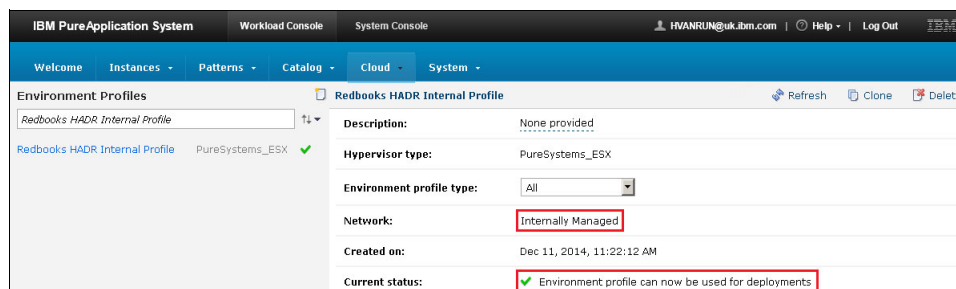


Figure 5-29 Environment Profile that supports deployments within a single system.

To deploy a specific pattern, use the following steps:

1. From the Workload Console, go the **Patterns** → **Virtual System Patterns** and filter for itsq. Then find the pattern ITSO DB2 OLTP HADR Pattern.
2. Click the **Deploy** icon as shown in Figure 5-30.

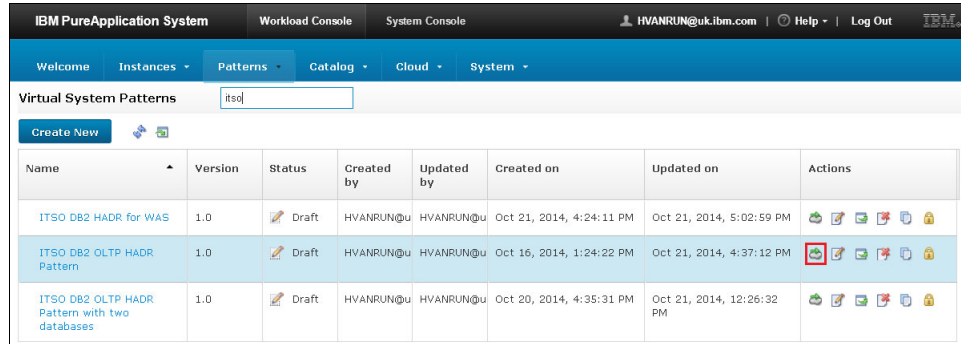


Figure 5-30 Deploying Virtual System Pattern, ITSO DB2 OLTP HADR Pattern

3. The deployment window opens in a new browser tab. Use the following steps to address the various parameters and attributes:
 - a. Specify the following generic deployment parameters:
 - Name
 - Environment profile
 - Priority
 - SSH key

Note: Make sure to specify the correct environment profile.

- b. Specify the Pattern and Component attributes. Table 5-5 provides more details about each attribute.

Table 5-5 Attributes for deployment of ITSO DB2 OLTP HADR Pattern

| Attribute | Type | Value |
|---------------------------|----------------------------------|-------|
| Password (root) | Pattern attribute | ***** |
| Password (virtuser) | Pattern attribute | ***** |
| Password (Fenced user) | Pattern attribute | ***** |
| Password (Instance owner) | Pattern attribute | ***** |
| databaseUserPassword | Pattern attribute | ***** |
| HADR sync mode | Component attribute (DB2 VM one) | SYNC |

After all the information has been entered, the window should look like Figure 5-31.

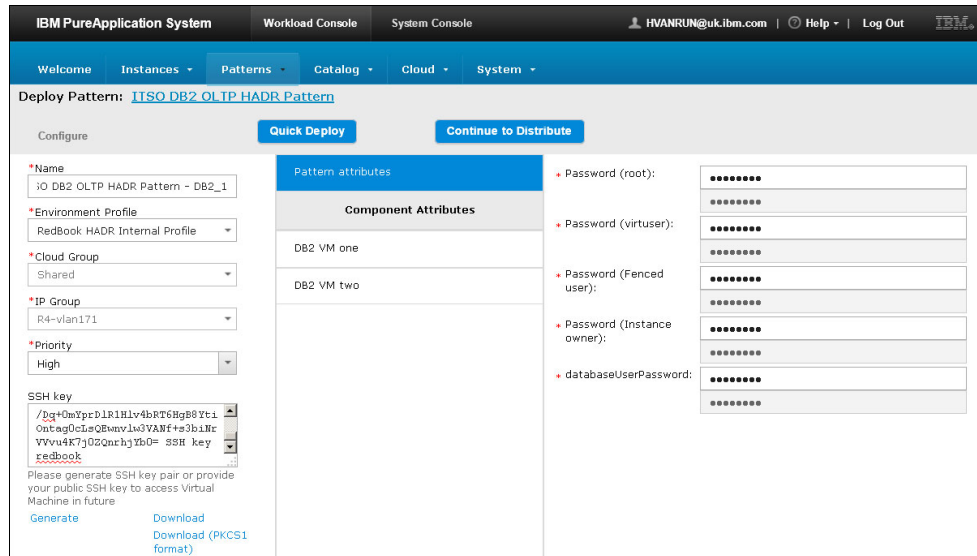


Figure 5-31 Deployment window for deployment of ITSO DB2 OLTP HADR Pattern

4. Click **Continue to Distribute**. The following message is shown while the system is preparing for this step (see Figure 5-32).

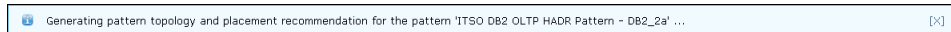


Figure 5-32 Verification message

5. The next window, shown in Figure 5-33, is where you assign the two different virtual machines to various Cloud Groups and IP Groups. However, in this example, the Environment Profile deploys to a single Cloud Group and IP Group. Notice that both virtual machines are assigned to Cloud Group Shared and IP Group R4-vlan171 in the PureApplication System (with serial number 1000202).

Option: If your Environment Profile contains only a single Cloud Group and IP Group, optionally at step 4, you can click **Quick Deploy** (instead of **Continue to Distribute**). This starts deployment immediately.

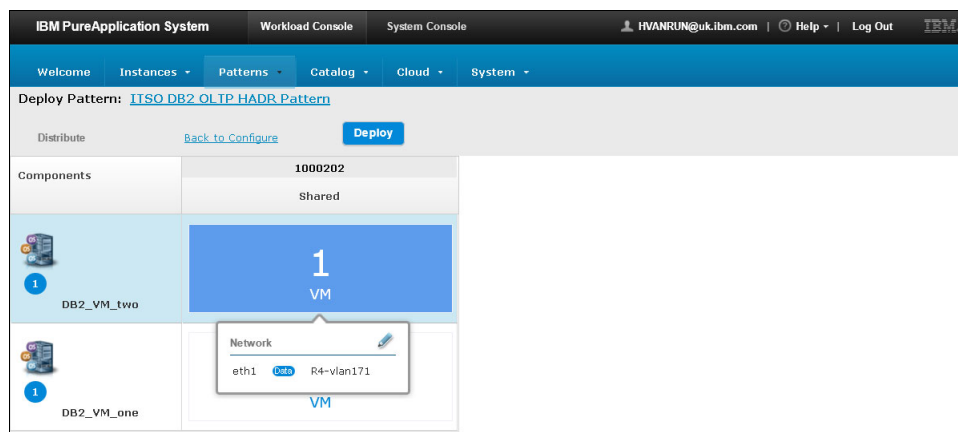


Figure 5-33 Accepting the distribution of the two virtual machines before deployment.

- Click **Deploy** to start the deployment. You can review the status by clicking the link provided (see Figure 5-34).

The instance 'ITSO DB2 OLTP HADR Pattern - DB2_1' is launching. You can check the status by [clicking here](#)

Figure 5-34 Link to status window

After the link is started, the deployment status window is displayed (see Figure 5-35).

| Name | Status | Created by | Created on | Actions |
|------------------------------------|-----------|--------------------|--------------------------|--|
| ITSO DB2 OLTP HADR Pattern - DB2_1 | Launching | HVANRUN@uk.ibm.com | Oct 22, 2014, 6:35:29 PM | [Refresh] [Start] [Stop] [Manage] [Maintain] [Resume] [Delete] |

Figure 5-35 Status of deployment

- After approximately 15 minutes, the deployment should have completed. Figure 5-36 shows the result of a successful deployment for a single PureApplication System.

| Name | Status |
|------------------------------------|---------|
| ITSO DB2 OLTP HADR Pattern - DB2_1 | Running |

| | |
|---|--|
| ITSO DB2 OLTP HADR Pattern - DB2_1 | [Refresh] [Start] [Stop] [Manage] [Maintain] [Resume] [Delete] |
| Created by | HVANRUN@uk.ibm.com |
| Created on | Oct 22, 2014, 6:35:29 PM |
| Access granted to | hvanrun@uk.ibm.com [all] Add more... |
| ID | d-3bb0a30f-11f6-4d03-81cd-3186e330c2be |
| Status | Running |
| Using environment profile | RedBook HADR Internal Profile |
| Priority | High |
| In cloud group | Shared |
| Referenced shared services | Red Hat Satellite Service |
| Pattern type | Virtual System Pattern Type 1.0 [Check for updates] |
| From pattern | ITSO DB2 OLTP HADR Pattern |
| Snapshots | |
| Middleware perspective (22 in total) | |
| DB2_Server_one-Part (DB2_VM_one) | |
| DB2_VM_one-Image (DB2_VM_one) | |
| primary_database_-_ITSO-Script (DB2_VM_one) | |
| DB2_Server_two-Part (DB2_VM_two) | |
| DB2_VM_two-Image (DB2_VM_two) | |
| standby_database_-_ITSO-Script (DB2_VM_two) | |

Figure 5-36 Successful deployment of ITSO DB2 OLTP HADR Pattern for a single system

5.4.2 Validation

See 5.7, “Validation” on page 160, for more details about how to prove that the deployed solution does actually provide HADR for the deployed DB2 database.

5.5 Scenario DB2_2: Two systems using a single pattern

This scenario deploys a DB2 primary and standby database across two PureApplication Systems in the PDC. Figure 5-37 shows the scenario, where DB2-P stands for the primary database and DB2-S for the standby database.

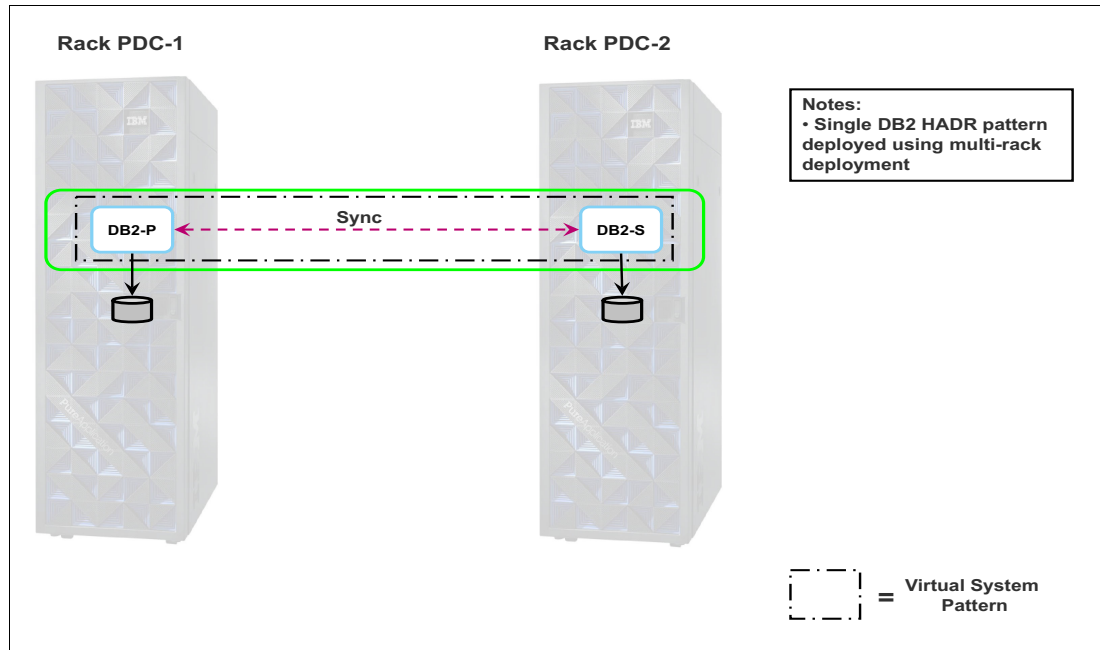


Figure 5-37 Scenario DB2_2: DB2 primary and standby using two different systems for the PDC

5.5.1 Deployment

This scenario deploys the Virtual System Pattern, ITSO DB2 OLTP HADR Pattern. See 5.3, “Building a DB2 Virtual System Pattern” on page 93 for more detail about this Virtual System Pattern. Figure 5-38 shows the ITSO DB2 OLTP HADR Pattern. Note that block storage is not used in this pattern.

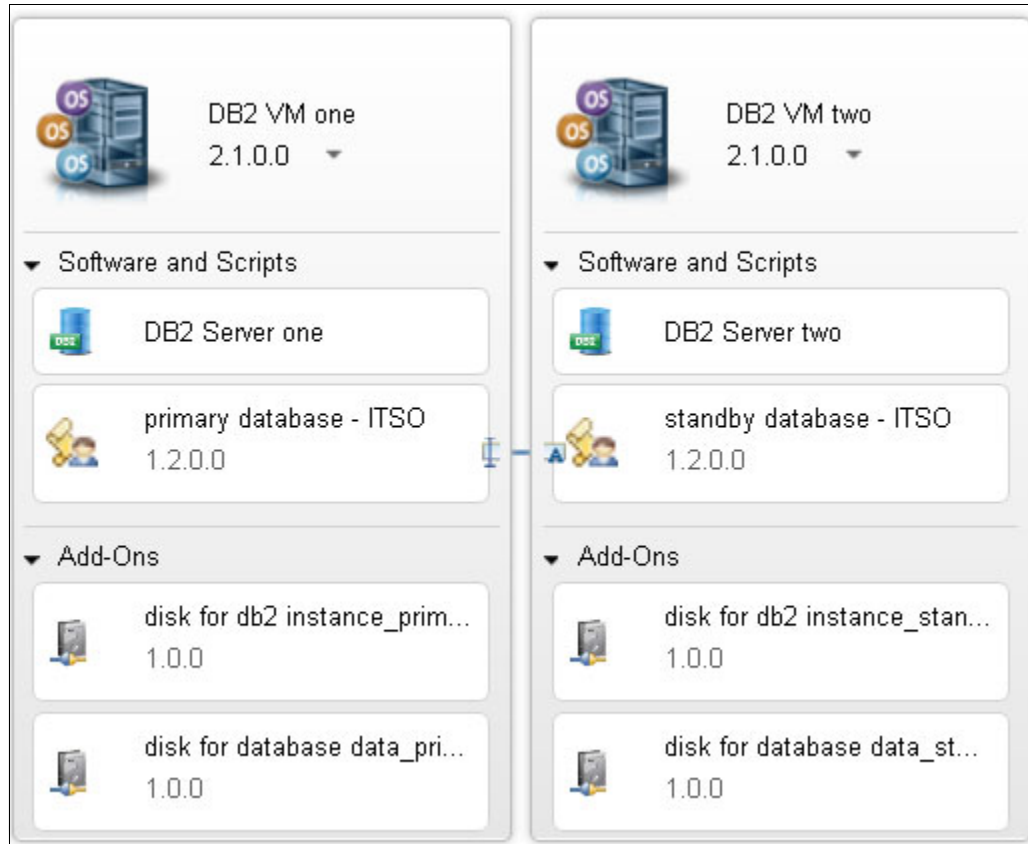


Figure 5-38 Virtual System Pattern, ITSO DB2 OLTP HADR Pattern

This scenario assumes that you have set up the two PureApplication Systems in the PDC to support *multitarget deployment* as described in Chapter 4, “Infrastructure setup” on page 47. For this deployment, use the Environment Profile Redbooks HADR External, which supports multitarget deployment. This type of Environment Profile uses an *externally managed* network, as shown in Figure 5-39. The status of the Environment Profile must also be valid. If the status is valid, the Current Status field states Environment profile can now be used for deployments as shown in Figure 5-39.

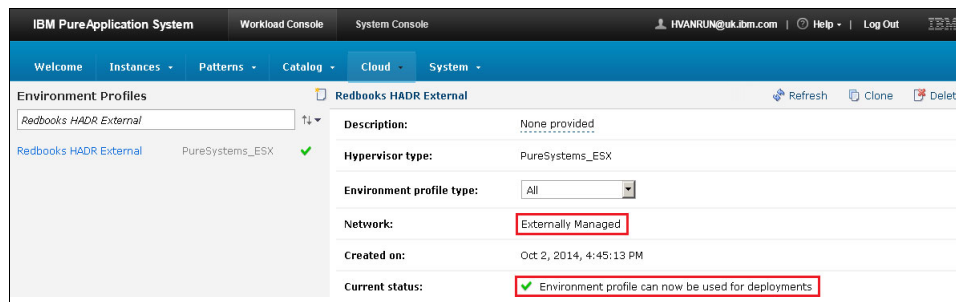


Figure 5-39 Environment Profile that supports multitarget deployments

To deploy the Virtual System Pattern ITSO DB2 OLTP HADR Pattern across two PureApplication Systems in the PDC, use the following steps:

1. From the Workload Console, go to **Patterns** → **Virtual System Patterns** and filter for its0.
2. Find the pattern ITSO DB2 OLTP HADR Pattern and click the **Deploy** icon (see Figure 5-40).

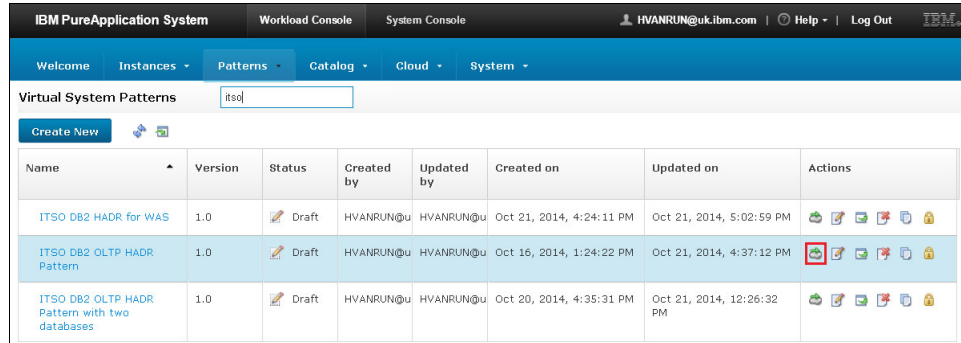


Figure 5-40 Deploying Virtual System Pattern ITSO DB2 OLTP HADR Pattern

3. The deployment window opens in a new browser tab. Within that window, use the following steps to specify parameters and attributes:
 - a. Specify the following generic deployment parameters:
 - Name
 - Environment Profile
 - Priority
 - SSH key

Note: Make sure to specify the multi-rack Environment Profile.

- b. In addition, you must specify Pattern and Component attributes. Table 5-6 provides more details about these attributes.

Table 5-6 Attributes for deployment of ITSO DB2 OLTP HADR Pattern

| Attribute | Type | Value |
|---------------------------|----------------------------------|-------|
| Password (root) | Pattern attribute | ***** |
| Password (virtuser) | Pattern attribute | ***** |
| Password (Fenced user) | Pattern attribute | ***** |
| Password (Instance owner) | Pattern attribute | ***** |
| databaseUserPassword | Pattern attribute | ***** |
| HADR sync mode | Component attribute (DB2 VM one) | SYNC |

After all information has been entered, the deployment window should look like Figure 5-41.

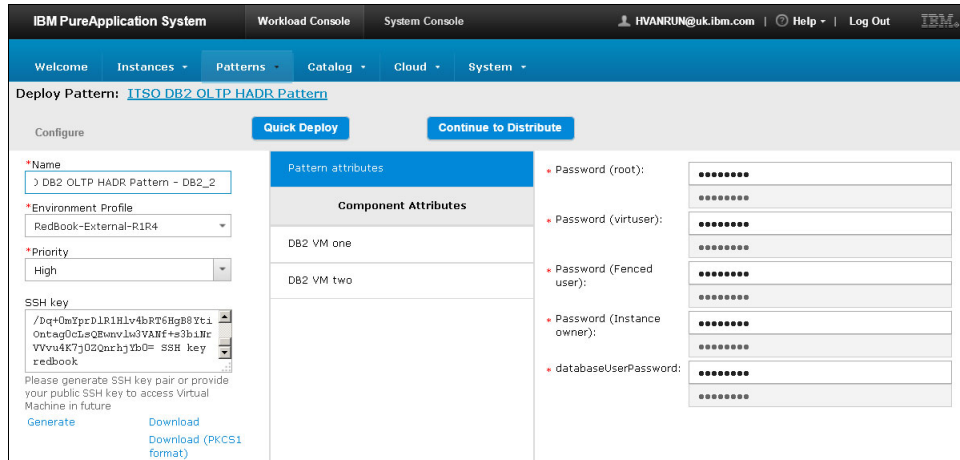


Figure 5-41 Deployment window for deployment of ITSO DB2 OLTP HADR Pattern

4. Click **Continue to Distribute**. The message in Figure 5-42 is shown while the system is preparing for this step.

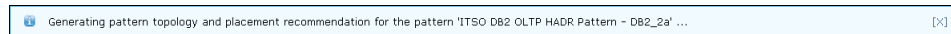


Figure 5-42 Verification message

5. The next window, shown in Figure 5-43, is where you assign the two different virtual machines to various Cloud Groups within both PureApplication Systems. By default, one virtual machine is assigned to each PureApplication System. In this example, the systems (with serial numbers 1000202 and 1000236) are used.

Note: By hovering the mouse over each virtual machine, you can review the IP Groups for the *Data* and *Management* networks.

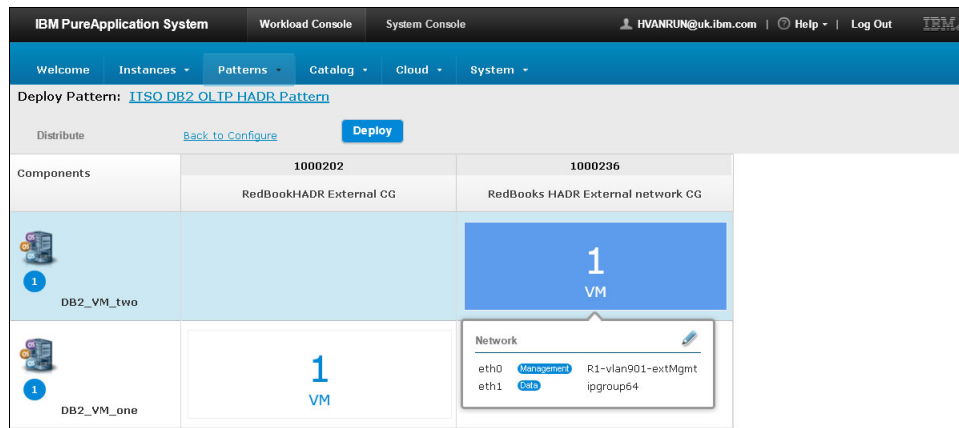


Figure 5-43 Distributing the two virtual machines across both PureApplication Systems

- Click **Deploy** to start the deployment. You can review the status by clicking the link provided (see Figure 5-44).

The instance 'ITSO DB2 OLTP HADR Pattern - DB2_2' is launching. You can check the status by [clicking here](#) [X]

Figure 5-44 Clickable link to review status of deployment

After you click the link for checking the status, the status window displays (see Figure 5-45).

| Name | Status | Created by | Created on | Actions |
|------------------------------------|-----------|--------------------|---------------------------|--|
| ITSO DB2 OLTP HADR Pattern - DB2_2 | Launching | HVANRUN@uk.ibm.com | Oct 22, 2014, 11:30:35 AM | [Refresh] [Start] [Stop] [Manage] [Maintain] [Resume] [Delete] |

Figure 5-45 Status window for deployment

- After approximately 15 minutes, the deployment completes. Figure 5-46 shows the result of a successful deployment across two PureApplication Systems using multi-rack.

ITSO DB2 OLTP HADR Pattern - DB2_2

Refresh Start Stop Manage Maintain Resume Delete

Created by: HVANRUN@uk.ibm.com

Created on: Oct 22, 2014, 11:30:35 AM

Access granted to: hvanrun@uk.ibm.com [all]

ID: d-84d20d8e-a885-4758-95e0-d468f4c80280

Status: Running

Using environment profile: RedBook-External-R1R4

Priority: High

In cloud group: RedBookHADR External CG @ Local
RedBooks HADR External network CG @ 1000236

Referenced shared services: Red Hat Satellite Service

Pattern type: Virtual System Pattern Type 1.0 [Check for updates]

From pattern: ITSO DB2 OLTP HADR Pattern

Snapshots

Middleware perspective (22 in total)

- DB2_Server_one-Part (DB2_VM_one)
- DB2_VM_one-Image (DB2_VM_one)
- primary_database_-_ITSO-Script (DB2_VM_one)
- DB2_Server_two-Part (DB2_VM_two)
- DB2_VM_two-Image (DB2_VM_two)
- standby_database_-_ITSO-Script (DB2_VM_two)

Show more

Figure 5-46 Successful deployment of ITSO DB2 OLTP HADR Pattern using multi-rack.

5.5.2 Validation

See 5.7, “Validation” on page 160 for more details about how to prove that the deployed solution provided HADR for the deployed DB2 database.

5.6 Scenario DB2_3: Two systems using block storage replication

Not all clients have a multi-rack setup in place. However, even without a multi-rack setup, it is still possible to deploy DB2 across two PureApplication Systems. Ultimately, all that the DB2 middleware requires for HADR is network connectivity between the two virtual machines running the DB2 instances with the primary and standby database (see Figure 5-47).

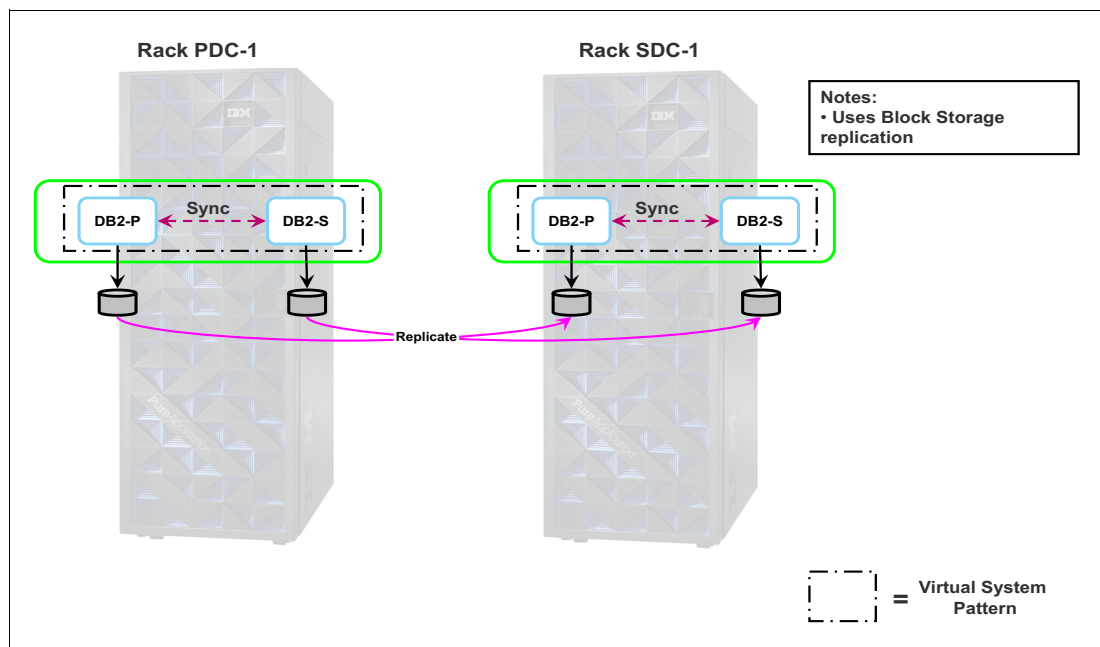


Figure 5-47 Scenario DB2_3: DB2 primary and standby using two different systems for the PDC

To deploy this scenario, you need to build two separate patterns and deploy them to the two different PureApplication Systems in the PDC.

5.6.1 Adding block storage to Virtual System Pattern

Adding block storage requires you to build a new Virtual System Pattern that uses Block Storage Volumes instead of normal Storage Volumes (which cannot be replicated between systems). To create a pattern by cloning the existing pattern ITS0 DB2 OLTP HADR Pattern, use the following steps:

1. From the Workload Console, go to **Patterns** → **Virtual Systems**. Filter for ITS0 to find the pattern ITS0 DB2 OLTP HADR Pattern. Click **Clone** to clone the pattern (see Figure 5-48).

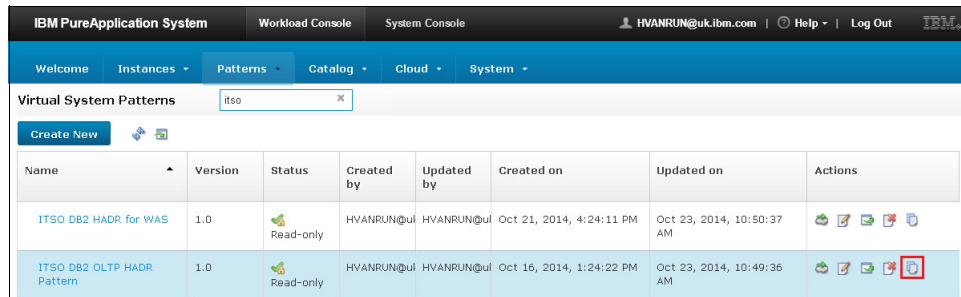


Figure 5-48 Filter and clone a pattern

2. Enter ITS0 DB2 OLTP HADR Pattern with BS, in the name field. Verify that you select the same virtual image as in the original pattern, which is listed on the right side under *Virtual images in the pattern*.(see Figure 5-49).

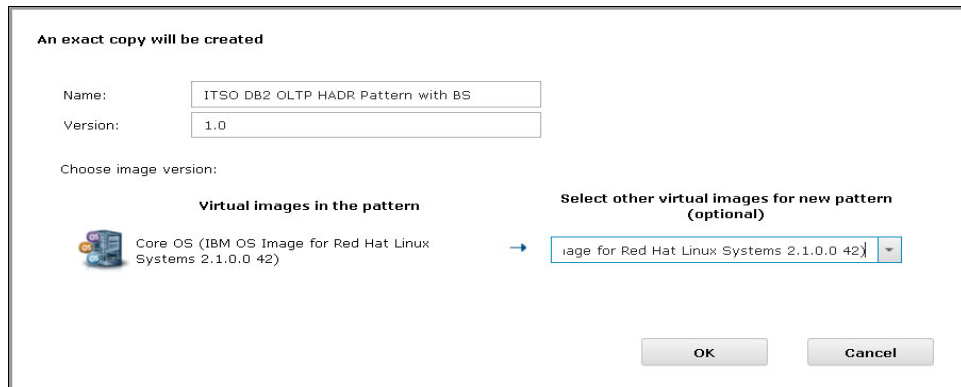


Figure 5-49 Configuring the clone

3. Find the new pattern, ITS0 DB2 OLTP HADR Pattern with BS, in the list. Click the **Edit** icon to open the pattern in the Pattern Builder (see Figure 5-50).

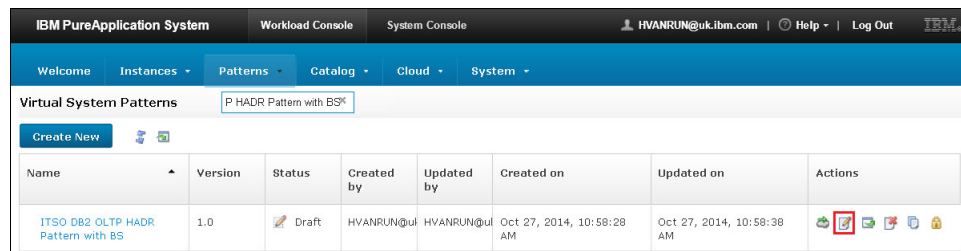


Figure 5-50 Finding the pattern and clicking the edit icon

- Changing the pattern can be done through the graphical editor. However, this example makes the changes by directly editing the corresponding source representation of the pattern. In the Pattern Builder, click **Source**, to open the Source window. Click **Edit** to start editing the source code for the pattern (see Figure 5-51).

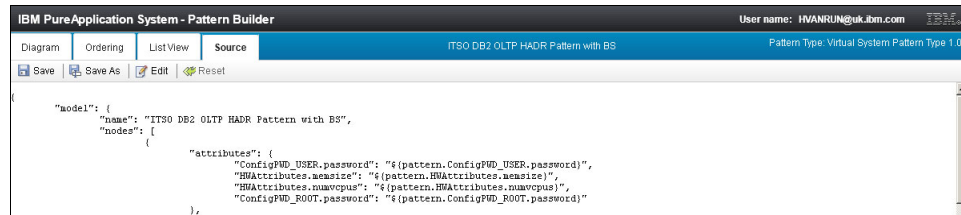


Figure 5-51 Opening the source window to edit the pattern

- Find the section in the source that has the four disk add-ons (corresponding to what is shown in Example 5-1). You should see, four disk add-ons of type `add-on:Default add disk:1.0.0`. Each of the two VMs has two of those disks that are associated with it. One is associated with the DB2 instance (mounted at `/myinst`) and the other one for the actual DB2 data (mounted at `/db2data`).

Example 5-1 The section for the four disk add-ons in the Virtual System Pattern source

```
{
  "attributes": {
    "VOLUME_GROUP": "group1",
    "DISK_SIZE_GB": 10,
    "FILESYSTEM_TYPE": "ext3",
    "OWNER": "root",
    "MOUNT_POINT": "/myinst"
  },
  "id": "disk for db2 instance_primary",
  "type": "add-on:Default add disk:1.0.0",
  "groups": {},
  "locked": [
    "DISK_SIZE_GB",
    "FILESYSTEM_TYPE",
    "MOUNT_POINT",
    "VOLUME_GROUP",
    "OWNER"
  ]
},
{
  "attributes": {
    "VOLUME_GROUP": "group2",
    "DISK_SIZE_GB": 30,
    "FILESYSTEM_TYPE": "ext3",
    "OWNER": "root",
    "MOUNT_POINT": "/db2data"
  },
  "id": "disk for database data_primary",
  "type": "add-on:Default add disk:1.0.0",
  "groups": {},
  "locked": [
    "DISK_SIZE_GB",
    "FILESYSTEM_TYPE",
    "MOUNT_POINT",

```

```

        "VOLUME_GROUP",
        "OWNER"
    ]
},
{
    "attributes": {
        "VOLUME_GROUP": "group1",
        "DISK_SIZE_GB": 10,
        "FILESYSTEM_TYPE": "ext3",
        "OWNER": "root",
        "MOUNT_POINT": "/myinst"
    },
    "id": "disk for db2 instance_standby",
    "type": "add-on:Default add disk:1.0.0",
    "groups": {},
    "locked": [
        "DISK_SIZE_GB",
        "FILESYSTEM_TYPE",
        "MOUNT_POINT",
        "VOLUME_GROUP",
        "OWNER"
    ]
},
{
    "attributes": {
        "VOLUME_GROUP": "group2",
        "DISK_SIZE_GB": 30,
        "FILESYSTEM_TYPE": "ext3",
        "OWNER": "root",
        "MOUNT_POINT": "/db2data"
    },
    "id": "disk for database data_standby",
    "type": "add-on:Default add disk:1.0.0",
    "groups": {},
    "locked": [
        "DISK_SIZE_GB",
        "FILESYSTEM_TYPE",
        "MOUNT_POINT",
        "VOLUME_GROUP",
        "OWNER"
    ]
},
}

```

-
6. For the primary and standby databases, change the type of disk from add-on:Default add disk:1.0.0 to add-on:Default attach block disk:1.0.0. This change effectively replaces the old disks with block storage disks. Change the type of disk for the following databases:
- ID: disk for database _primary
 - ID: disk for database _standby

The sections, in bold in Example 5-2, show the source of the pattern after making these changes.

Example 5-2 Pattern source with block storage disks for the databases incorporated

```
{
  "attributes": {
    "VOLUME_GROUP": "group1",
    "DISK_SIZE_GB": 10,
    "FILESYSTEM_TYPE": "ext3",
    "OWNER": "root",
    "MOUNT_POINT": "/myinst"
  },
  "id": "disk for db2 instance_primary",
  "type": "add-on:Default add disk:1.0.0",
  "groups": {},
  "locked": [
    "DISK_SIZE_GB",
    "FILESYSTEM_TYPE",
    "MOUNT_POINT",
    "VOLUME_GROUP",
    "OWNER"
  ]
},
{
  "attributes": {
    "FILESYSTEM_TYPE": "ext3",
    "MOUNT_POINT": "/db2data"
},
  "id": "disk for database data_primary",
  "type": "add-on:Default attach block disk:1.0.0",
  "groups": {}
},
{
  "attributes": {
    "VOLUME_GROUP": "group1",
    "DISK_SIZE_GB": 10,
    "FILESYSTEM_TYPE": "ext3",
    "OWNER": "root",
    "MOUNT_POINT": "/myinst"
  },
  "id": "disk for db2 instance_standby",
  "type": "add-on:Default add disk:1.0.0",
  "groups": {},
  "locked": [
    "DISK_SIZE_GB",
    "FILESYSTEM_TYPE",
    "MOUNT_POINT",
    "VOLUME_GROUP",
    "OWNER"
  ]
},
{
  "attributes": {
    "FILESYSTEM_TYPE": "ext3",
    "MOUNT_POINT": "/db2data"

```

```

    },
    "id": "disk for database data_standby",
    "type": "add-on:Default attach block disk:1.0.0",
    "groups": {}
  },

```

7. Click **Done Editing** to finish editing the source of the Virtual System Pattern. This action automatically saves the pattern.

Note: The new pattern, when viewed in the Pattern Builder using the (default) graphical interface Diagram tab, does not show any difference between the normal disk add-ons and the Block Storage disk add-ons. However, the difference can be seen by inspecting the parameters that they expose in the Pattern Builder.

The new pattern, ITS0 DB2 OLTP HADR Pattern with BS, created for the system in the PDC, also needs to be made available for the SDC. See the following links for details about how to export and import Virtual System Patterns:

http://www.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/iwd/pat_exportvsys.dita

http://www.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/iwd/pat_importvsys.dita

5.6.2 Block Storage configuration

Before deploying the new pattern, you need to define Block Storage Volumes on both systems. Table 5-7 shows the different Storage Volumes that you need to create. These Storage Volumes also need to be associated with a new Storage Group Volume on each system.

Storage Volume Groups can be used to logically group a number of Storage Volumes. As the Storage Volumes you create are for a single purpose, this is a useful opportunity for grouping (also included in Table 5-7).

Table 5-7 Block Storage Volumes and their attributes as configured for this scenario

| Storage Volume | Size (GB) | System | Cloud Group | Storage Volume Group |
|--------------------|-----------|--------|-------------|----------------------|
| DB2_3 VM one - PDC | 10 | PDC | Shared | DB2_3 |
| DB2_3 VM two - PDC | 10 | PDC | Shared | DB2_3 |
| DB2_3 VM one - SDC | 10 | SDC | Shared | DB2_3 |
| DB2_3 VM one - SDC | 10 | SDC | Shared | DB2_3 |

Define Storage Volume Groups

To define Storage Volume Groups, complete the following steps:

1. Log on to the System Console of the system in the PDC and select **Cloud** → **Storage Volume Groups**. Then click **Create New** (see Figure 5-52).

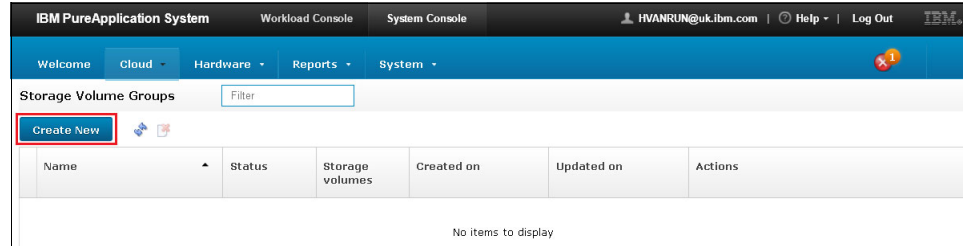


Figure 5-52 Create a Storage Volume Group

2. Specify the following field attributes for the new Storage Volume Group (see Figure 5-53):
 - a. Name: DB2_3
 - b. Description: Provide a useful description for the Storage Volume Group, this example uses Storage volume group for scenario DB2_3
 - c. Cloud Group: Specify the name of the Cloud Group where you are deploying the pattern to; this example uses the Cloud Group Shared

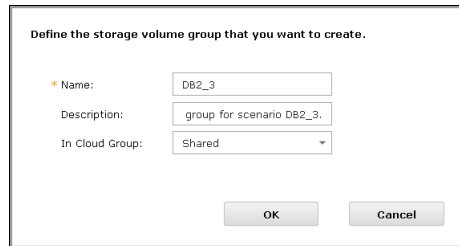


Figure 5-53

3. Click **OK**, to create the Storage Volume Group.
4. Repeat these steps to create the same Storage Volume Group on the system in the SDC.

Create Block Storage Volumes

With the Storage Volume Groups created, you can now create the Block Storage Volumes and associate them with the appropriate Storage Volume Groups. To do so, complete the following steps:

1. From the System Console of the system in the PDC, select **Cloud** → **Storage Volumes**. Then click **Create New** (see Figure 5-54).

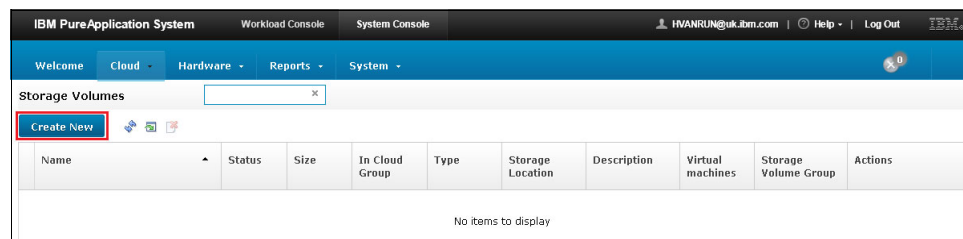


Figure 5-54 Creating Block Storage volumes

2. Specify the following field attributes for the new Storage Volume (see Figure 5-55):
 - a. Name: DB2_3 VM one - PDC
 - b. Description: Provide a useful description for the Storage Volume, this example uses Block storage volume of VM one in PDC
 - c. Type: Make sure to choose Block; this selection must be a Block Storage Volume to be able to replicate it.
 - d. Storage Volume Group: Select the Storage Volume Group, DB2_3, that you created previously.
 - e. Cloud Group: Select the Cloud Group that will be used for the deployment in the PDC. In this example, use the Cloud Group, Shared.
 - f. Size: Specify 10 GB for the size of the new Storage Volume

The screenshot shows a dialog box titled "Create a new storage volume". It contains the following fields and values:

- Name: DB2_3 VM one - PDC
- Description: volume of VM one in PDC
- Type: Block
- Storage Volume Group: DB2_3
- In Cloud Group: Shared

Volume configuration: Existing settings (unselected), Customize settings (selected)

Size: 10 GB

Buttons: OK, Cancel

Figure 5-55 Specifying the attributes for the new Storage Volume

3. Click **OK** to create it.
4. Create the second Block Storage Volume on the system in the PDC, and the two Block Storage Volumes on the system in the SDC in a similar fashion. See Table 5-7 on page 124 for the information that you need to provide.

Figure 5-56 shows the Storage Volume Group, DB2_3, and its two Block Storage Volumes on the system in the PDC.

Note: Replication type for the two Block Storage Volumes is listed as Asynchronous. This is the default. After you set up synchronous replication, the information is updated to reflect the change.

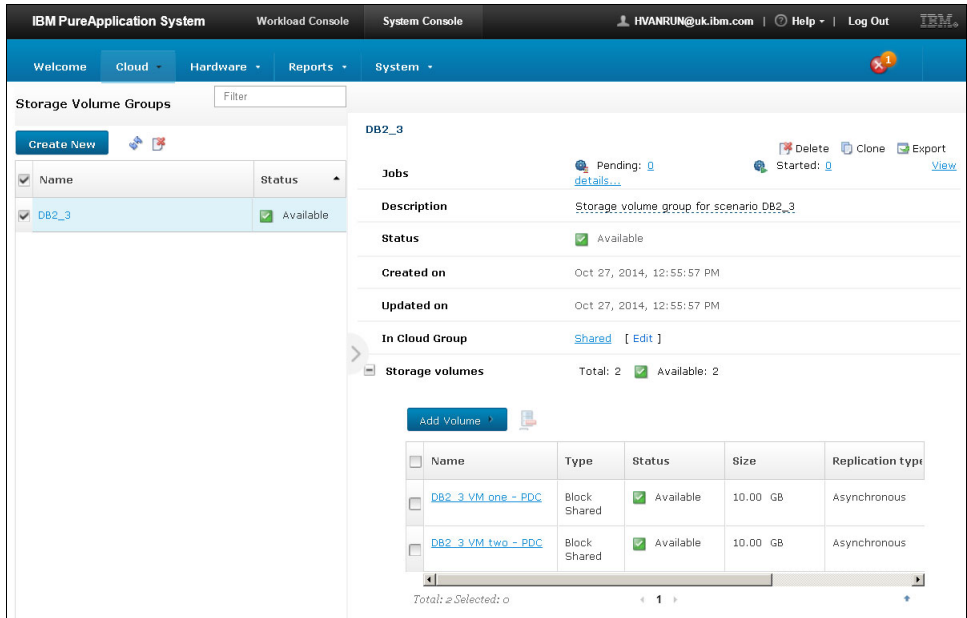


Figure 5-56 Block Storage Volume Group, DB2_3, on the system in the PDC

5.6.3 Deploy Virtual System Pattern on both systems

With the Block Storage Volumes defined in both systems, you can now deploy the Virtual System Pattern. To do so, complete the following steps:

1. Go to the Workload Console of the system in the PDC and select **Patterns** → **Virtual Systems**.
2. Find the pattern, ITS0 DB2 OLTP HADR Pattern with BS, and click the **Deploy** icon (see Figure 5-57).

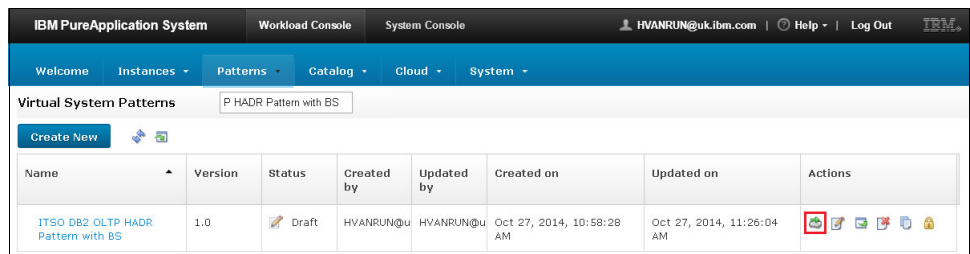


Figure 5-57 Deploying the Virtual System Pattern

3. Several options are located to the left in the Deploy Pattern window. Specify the following listed options and accept the default information for options not listed (see Figure 5-58):
 - a. Name: Provide a unique name, for this example use, ITSO DB2 OLTP HADR Pattern with BS - DB2_3
 - b. Environment Profile: Choose the appropriate Environment Profile for the system in the PDC. For this example, use Redbooks HADR Internal Profile.

Note: The Cloud Group, Shared, matches the Cloud Group that you specified when creating the Block Storage Volumes on the system in the PDC.

- c. SSH key: Although not required, it can be convenient to specify an SSH key for access to the deployed VMs.
- d. Storage Volume Group: Choose the Storage Volume Group, DB2_3, that you created previously.
- e. Cloud Group: Choose the Cloud Group that will be used for the deployment in the PDC. For this example, use the Cloud Group, Shared.
- f. Size: Specify 10 GB for the size of the new Storage Volume.

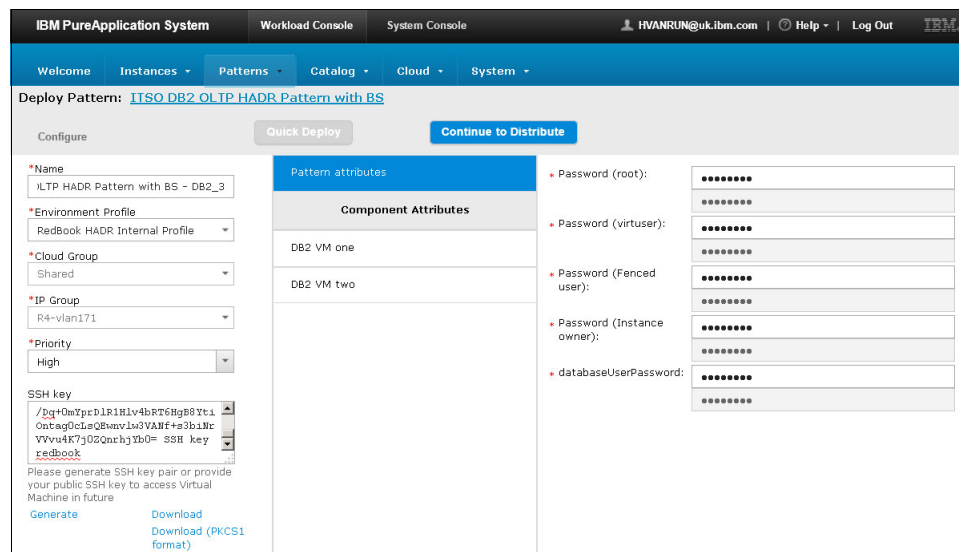


Figure 5-58 Configuring pattern options and fields

4. Under Pattern attributes (located to the right), specify a password for the following users:
 - root
 - virtuser
 - Fenced user (db2fenc1)
 - Instance owner (db2inst1)
 - Database user (itsouser)

- Under Component attributes (located to the right), select **DB2 VM one** and expand the attributes for primary database - ITS0. For the HADR Sync Mode, select **SYNC** (see Figure 5-59).

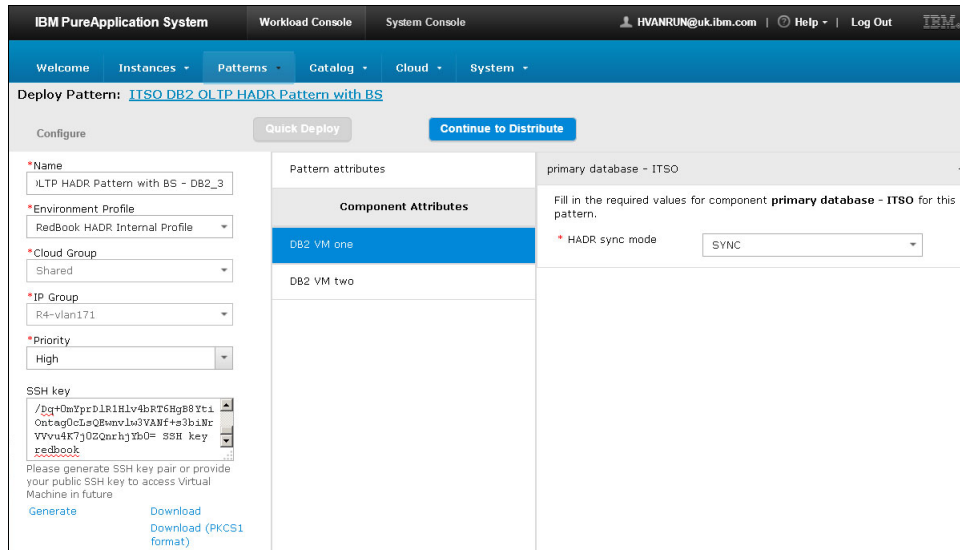


Figure 5-59 Configuring attributes and fields for the pattern before deploying

- Click **Continue to Distribute**.

Note: The **Quick Deploy** option is grayed out because this pattern contains Block Storage Volumes. At deployment time, you must specify whether to use existing Block Storage Volumes or to create new ones. This example uses existing block storage volumes.

- Using the Distribute deployment window, hover the mouse over **DB2_VM_one**. Notice that no Storage Volume has been associated yet for the primary database known as `disk_for_database_data_primary` (see Figure 5-60).

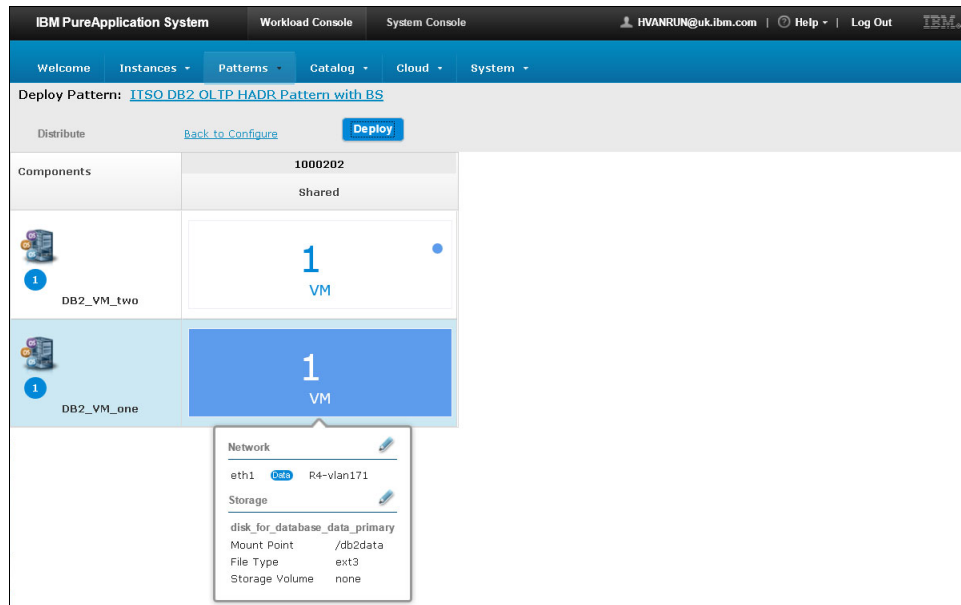


Figure 5-60 Viewing associated Storage Volume for the primary database

- To associate a storage volume, click the **Edit** icon for the Storage of **DB2_VM_one**. In the window that opens, you see the drop-down menu for the Storage Volume. Select **DB2_3 VM one - PDC** (see Figure 5-61).

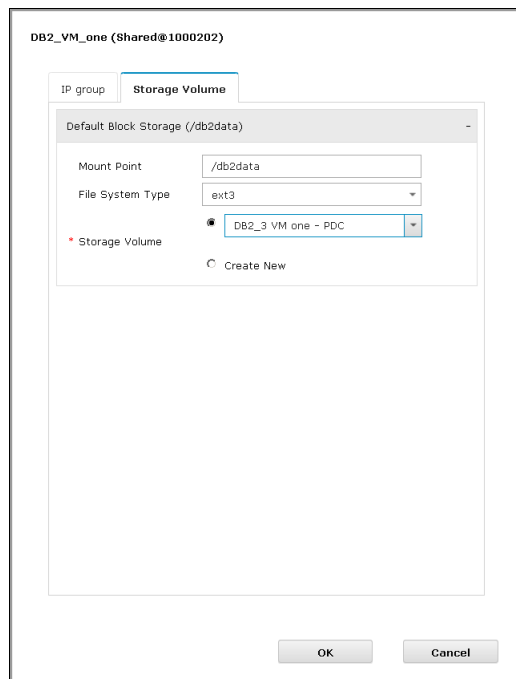


Figure 5-61 Select a Storage Volume for DB2_VM_one

- Repeat steps 7 on page 130 and 8 on page 130 for DB2_VM_two, but select **DB2_3 VM two - PDC** as the Storage volume (see Figure 5-62).

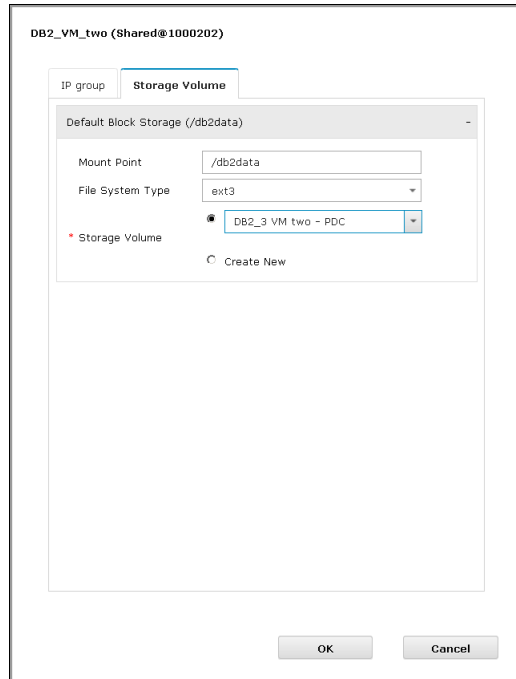


Figure 5-62 Select a Storage Volume for DB2_VM_two

- You are now ready to deploy the pattern. Click **Deploy** to start the deployment. After a few moments, you should see a notification that the new Virtual System Instance is starting. Click the **link** to go to the Virtual System Instance and monitor the progress of its deployment (see Figure 5-63).

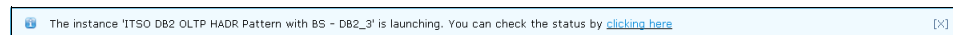


Figure 5-63 Clickable link to verification and monitoring of deployment

- After approximately 15 minutes, the deployment of the new Virtual System Instance should have completed (see Figure 5-64).

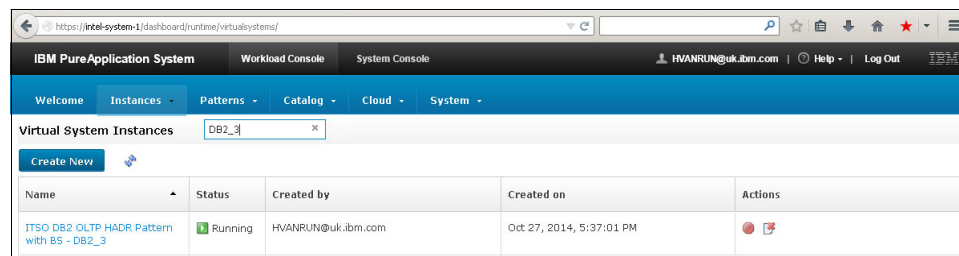


Figure 5-64 Verify that Virtual System Instance has completed deployment

12. Repeat steps 1 on page 127 through 11 on page 131 to deploy the same pattern on the system in the SDC. After completion, you should have a running Virtual System Instance for the SDC (see Figure 5-65).

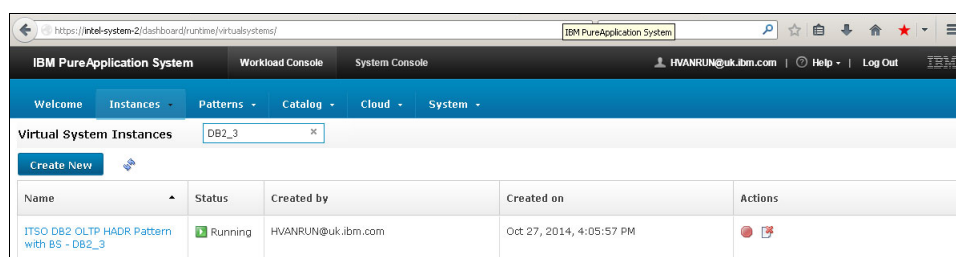


Figure 5-65 Verify that the Virtual System Instance has deployed for the SDC

5.6.4 Enable Block Storage replication

The DB2 HADR Virtual System Pattern is now running in both the PDC and SDC. Each system has two Block Storage Volumes that hold the DB2 database data (for example /db2data) for both the primary and standby DB2 VMs. Currently, these two Virtual System Instances are running independently.

Next, make the necessary changes to start replicating the data of the Block Storage Volumes from the PDC to the SDC. Block Storage Replication in PureApplication System mandates that only the source Block Storage Volume can be attached to a VM, the target Block Storage Volume must be detached.

Therefore, detach the two Block Storage Volumes from the Virtual System Instance in the SDC. But before you take that step, you must stop the DB2 instances running on the virtual machines of that Virtual System Instance.

Note: It is currently not possible to simply stop the Virtual System Instance in the SDC. In IBM PureApplication System 2.0.0.0, Block Storage Volumes can only be detached from *running* VMs. Therefore, you need to manually stop the DB2 instances and then detach the Block Storage Volumes.

Understanding Tivoli Storage Automation setup

Before you stop DB2 on the Virtual System Instance in the SDC, notice that the IBM Tivoli® System Automation has been automatically configured by the pattern. As a result, a Tivoli System Automation domain and a number of Resource Groups have been put in place.

The `lsrpdomain` command lists the domains. You can run this command from either of the VMs in the Virtual System Instance. Example 5-3 shows the normal output of this command.

Example 5-3 The Tivoli System Automation domain, db2HADomain, that is automatically set up

```
-bash-4.1# lsrpdomain
Name           OpState  RSCTActiveVersion MixedVersions TSPort  GSPort
db2HADomain   Online   3.1.4.4                No           12347  12348
```


The `lssam` command can be used to obtain the status of the Resource Groups that exist within the domain. Example 5-4 shows the output of this command. This command can be run from either VM.

Example 5-4 The status of the various Tivoli System Automation Resource Groups

```
-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
        '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Online IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs
        '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
        |- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
    |- Online IBM.PeerNode:ausipas088:ausipas088
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth1:ausipas089
    '- Online IBM.NetworkInterface:eth1:ausipas088
```

Table 5-8 lists the three Resource Groups and explains their purposes. There is a Resource Group corresponding to the DB2 instance on each of the two VMs in the Virtual System Instance.

Table 5-8 Overview of the three Resource Groups

| Resource group | Description | Expected status |
|-------------------------------|---|-----------------|
| db2_db2inst1_ausipas088_0-rg | DB2 instance “db2inst1” on host “ausipas088” | Online |
| db2_db2inst1_ausipas089_0-rg | DB2 instance “db2inst1” on host “ausipas089” | Online |
| db2_db2inst1_db2inst1_ITS0-rg | Controls which database is the primary and standby. | Online |

The Resource Group `db2_db2inst1_db2inst1_ITS0-rg` controls which “ITS0” database is the primary and standby. The output of the command `lssam` also shows that the Application, `db2_db2inst1_ITS0-rs`, is only *Online* on host `ausipas088`. This indicates that the primary ITS0 database is running on `ausipas088`. This is confirmed by issuing the command, `db2pd -hadr -db itso`, and examining the field called `HADR_ROLE`, as shown in Example 5-5.

Example 5-5 ITS0 database on ausipas088 is the primary

```
[db2inst1@ausipas088 ~]$ db2pd -hadr -db itso

Database Member 0 -- Database ITS0 -- Active -- Up 0 days 04:51:44 -- Date
2014-11-04-13.02.47.705287
```

```

      HADR_ROLE = PRIMARY
      REPLAY_TYPE = PHYSICAL
      HADR_SYNCMODE = SYNC
      STANDBY_ID = 1
      LOG_STREAM_ID = 0
      HADR_STATE = PEER
      HADR_FLAGS =
      PRIMARY_MEMBER_HOST = ausipas088
      PRIMARY_INSTANCE = db2inst1
      PRIMARY_MEMBER = 0
      STANDBY_MEMBER_HOST = ausipas089
      STANDBY_INSTANCE = db2inst1
      STANDBY_MEMBER = 0
      HADR_CONNECT_STATUS = CONNECTED
      HADR_CONNECT_STATUS_TIME = 11/04/2014 08:11:09.991581 (1415088669)
      HEARTBEAT_INTERVAL(seconds) = 30
      HADR_TIMEOUT(seconds) = 150
      TIME_SINCE_LAST_RECV(seconds) = 7
      PEER_WAIT_LIMIT(seconds) = 0
      LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000743
      LOG_HADR_WAIT_ACCUMULATED(seconds) = 10.681
      LOG_HADR_WAIT_COUNT = 9983
      SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 19800
      SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
      PRIMARY_LOG_FILE,PAGE,POS = S0000011.LOG, 563, 93141005
      STANDBY_LOG_FILE,PAGE,POS = S0000011.LOG, 563, 93141005
      HADR_LOG_GAP(bytes) = 0
      STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000011.LOG, 563, 93141005
      STANDBY_RECV_REPLAY_GAP(bytes) = 0
      PRIMARY_LOG_TIME = 11/04/2014 12:24:33.000000 (1415103873)
      STANDBY_LOG_TIME = 11/04/2014 12:24:33.000000 (1415103873)
      STANDBY_REPLAY_LOG_TIME = 11/04/2014 12:24:33.000000 (1415103873)
      STANDBY_RECV_BUF_SIZE(pages) = 4298
      STANDBY_RECV_BUF_PERCENT = 0
      STANDBY_SPOOL_LIMIT(pages) = 25600
      STANDBY_SPOOL_PERCENT = 0
      PEER_WINDOW(seconds) = 120
      PEER_WINDOW_END = 11/04/2014 13:04:40.000000 (1415106280)
      READS_ON_STANDBY_ENABLED = N

```

Stop DB2 on the Virtual System Instance in the SDC

After you review the information about the Tivoli System Automation Resource Group for the DB2 instance on each VM, you can stop DB2 on the Virtual System Instance in the SDC using Tivoli System Automation commands.

To stop the DB2 instance that is running the standby ITSO database, use the following steps:

1. Run the **chrg** command as shown in Example 5-6 to stop the corresponding Tivoli System Automation Resource Group from Table 5-8 on page 133.

Note: When running the **chrg** command to stop the Tivoli System Automation Resource Group, the command must be run as *root*.

Example 5-6 Stopping the Resource Group corresponding to the standby DB2 database

```
-bash-4.1# chrg -o Offline db2_db2inst1_ausipas089_0-rg
```

2. Run the **lssam** command and notice the Resource Group for the DB2 instance on host, *ausipas089*. Briefly, after running the command, the Resource Group becomes offline. This action is shown in Example 5-7.

Example 5-7 Confirming that the Resource Group for the standby DB2 database instance is offline.

```
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
    '- Online
IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Offline IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ausipas089_0-rs
    '- Offline
IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITSO-rg Request=Lock
Nominal=Online
  '- Online IBM.Application:db2_db2inst1_db2inst1_ITSO-rs
Control=SuspendedPropagated
  |- Online
IBM.Application:db2_db2inst1_db2inst1_ITSO-rs:ausipas088
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITSO-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITSO-rg_group-equ
  |- Online IBM.PeerNode:ausipas088:ausipas088
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth1:ausipas089
  '- Online IBM.NetworkInterface:eth1:ausipas088
```

3. Log on to the host, *ausipas089*, to determine the status of the local DB2 instance. Run the **db2pd** command. Example 5-8 shows that the **db2pd** command is unable to attach to the database manager, which implies that the local instance is not running.

Example 5-8 Confirming that the DB2 instance on the standby host is no longer running

```
[db2inst1@ausipas089 ~]$ db2pd -hadr -db itso
Unable to attach to database manager on member 0.
Please ensure the following are true:
  - db2start has been run for the member.
  - db2pd is being run on the same physical machine as the member.
  - DB2NODE environment variable setting is correct for the member
```

or `db2pd -mem` setting is correct for the member.
 Another possibility of this failure is the Virtual Address Space Randomization is currently enabled on this system.

4. With the DB2 instance on `ausipas089` stopped, there should be no open files in `/db2data`. The location (`/db2data`) is the block storage where the data for the ITSO database is stored. Run the `lsdf` command as shown in Example 5-9 to verify that there are no open files. If the command does not return any data, it means that there are no open files in `/db2data`.

Example 5-9 Confirm that no open files exist in /db2data

```
[db2inst1@ipas-lpar-9-3-171-24 ~]$ lsdf | grep /db2data
[db2inst1@ausipas089 ~]
```

5. Repeat steps 1 on page 135 through 4 to stop the DB2 instance on the primary database (VM `ausipas088`).

Detach Block Storage volumes from the VMs in SDC

Now that the DB2 instances on both VMs of the Virtual System Instance have been stopped, you can proceed with detaching the Block Storage volumes. For *both* VMs, use the following steps to detach their corresponding Block Storage Volume:

1. Log on to the VM and confirm that the file system `/db2data` is mounted. This file system is on the Block Storage Volume (which corresponds to `/dev/sdc1` on the VM). See Example 5-10.

Example 5-10 Confirm that the file system of the Block Storage Volume is mounted under /db2data

```
[db2inst1@ausipas089 ~]$ mount
/dev/mapper/vg_root-LogVol100 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/group1-LV9067 on /myinst type ext3 (rw)
/dev/sdc1 on /db2data type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

2. From the Virtual System Instance in the Workload Console, expand the **Virtual Machines** section that is on the right side of the window.
3. Expand the appropriate VM, and find the *Block storage* section. In that section, find the attached Block Storage Volume (mounted as an `ext3` file system on `/db2data`). Click **Detach** to detach the Block Storage Volume (see Figure 5-66).

| Block storage | | | | |
|---------------|-----------|--------------------|-------------------------------------|--------|
| Mount Point | File Type | Volume | Status | Action |
| /db2data | ext3 | DB2_3 VM one - PDC | <input checked="" type="checkbox"/> | Detach |

Notes: the virtual machine needs to be running in order to perform attach and detach operations.

Figure 5-66 Detaching the Block Storage Volume

4. After step 3 on page 136 is completed, the file system is no longer mounted. You can confirm this in the VM by again running the `mount` command (see Example 5-11).

Example 5-11 Confirm that the file system of the Block Storage Volume is no longer mounted

```
[db2inst1@ausipas089 ~]$ mount
/dev/mapper/vg_root-LogVol100 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/group1-LV9067 on /myinst type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

After both Block Storage Volumes have been detached, leave the Virtual System Instance running. This state allows for fast attach operations in a failover from the system in the PDC. However, the DB2 database instances are stopped and the data for the ITSO database is *not* available.

Configure Block Storage replication

The two Block Storage Volumes are now *detached* on the system in the SDC. That means you can now configure the replication of the *attached* Block Storage Volumes on the system in the PDC with those on the system in the SDC.

Before you proceed with configuring Block Storage Replication, ensure that you have set up a *Block Storage Replication Profile* on both systems. This is described in more detail in 4.2, “Block storage replication configuration” on page 49. On the systems in this scenario, a Block Storage Replication Profile called *Redbooks DR* is set up. Table 5-9 outlines the Block Storage Replication to configure for this scenario.

Table 5-9 Configuration for Block Storage Replication

| Block Storage Replication Profile | Source Block Storage Volume (PDC) | Target Block Storage Volume (SDC) |
|-----------------------------------|-----------------------------------|-----------------------------------|
| Redbooks DR | DB2 VM one - PDC | DB2 VM one - SDC |
| Redbooks DR | DB2 VM two - PDC | DB2 VM one - SDC |

To start the process of configuring for Block Storage Replication, use the following steps:

1. Go to the System Console of the system in the PDC and select **System** → **Block Storage Replication** (see Figure 5-67).



Figure 5-67 Selecting Block Storage Replication

2. Select the Block Storage Replication Profile, **Redbooks DR**. Click **Add Volumes** to start adding the Block Storage Volumes for this scenario (see Figure 5-68).

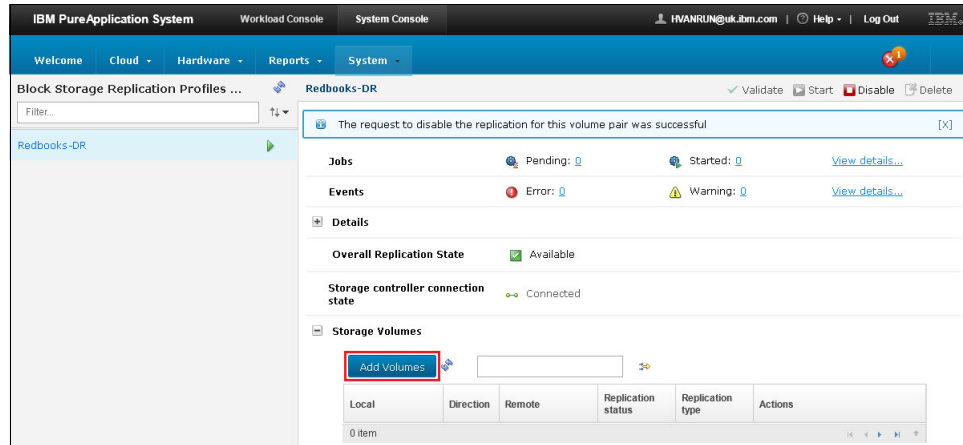


Figure 5-68 Adding volumes

3. Perform the following steps in the *Add storage volumes for replication* window:
 - a. Select **DB2_3 VM one - PDC** as the **Local Volume**.
 - b. After you select a Local Volume, the volumes that are listed under **Remote Volume** are populated with matching Block Storage Volumes on the system in the SDC. Under Remote Volume, select **DB2_3 VM one - SDC**.

- c. Select **Synchronous** as the Replication type, and then click **Add** (see Figure 5-69).

Figure 5-69 Configuring replication

- d. After adding volumes, you should see the following notification (see Figure 5-70).

Figure 5-70 Notification of success

- e. Click **Close**. You should see the Block Storage Volume, DB2_3 VM one - SDC, listed under Storage Volumes as shown in Figure 5-71. Note that you can still withdraw this request.

| Local | Direction | Remote | Replication status | Replication type | Actions |
|--------------------|-----------|--------------------|--------------------|------------------|----------|
| DB2_3 VM one - PDC | → | DB2_3 VM one - SDC | | Synchronous | Withdraw |

Figure 5-71 Listed Storage Volumes

- f. Next, log on to the system in the SDC and accept the request for replication of this Block Storage Volume. From the System Console, select **System** → **Block Storage Replication** and select the Block Storage Replication Profile, **Redbooks-DR** (see Figure 5-72).

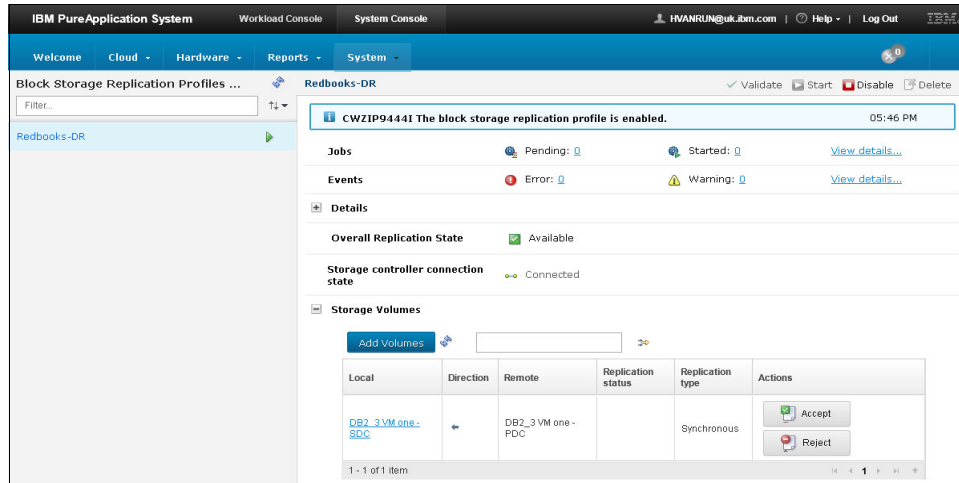


Figure 5-72 Selecting Block Storage Replication Profile

- g. Notice that DB2_3 VM one SDC is listed under Storage Volumes. This is the local Block Storage Volume on the system in the SDC. The remote Block Storage Volume is DB2_3 VM one PWC. This matches what is listed in Table 5-9 on page 137. Click **Accept** to start the replication (see Figure 5-73).

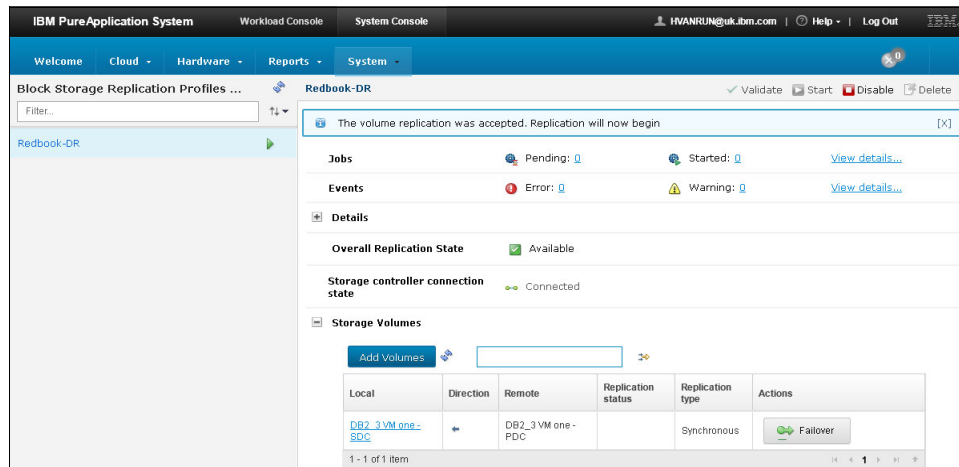


Figure 5-73 Notification that replication has started

4. Repeat step 3 on page 138 to add the second Block Storage Volume listed in Table 5-9 on page 137.

- When both Block Storage Volumes have been set up for replication, the Block Storage Replication Profile in the SDC should look as shown in Figure 5-74.

Note: The *Replication status* for both Block Storage Volumes is not shown. It takes some time until the initial synchronization is completed. Although typically the Fibre Channel connection between the two systems has a bandwidth of 16 Gbps, PureApplication System applies a rate limit of 64 MBps for this initial synchronization. Therefore, two Block Storage Volumes of 10 GB each, take approximately $2 * 160 = 320$ seconds (or 5 minutes).

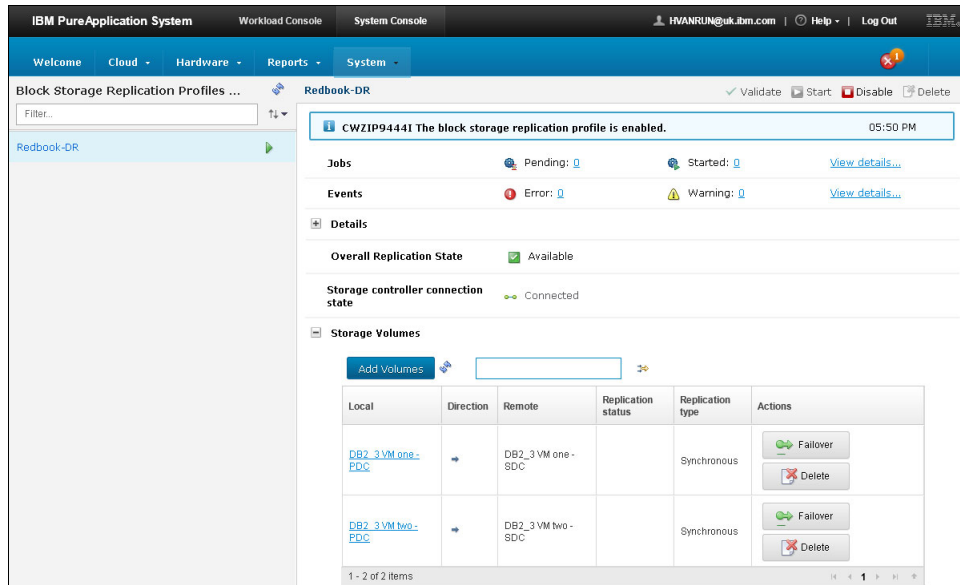


Figure 5-74 Block Storage Volume Replication seen from the system in the SDC

- After the initial synchronization has completed for both Block Storage Volumes, a Replication status of *Available* should be reported, as shown in Figure 5-75.

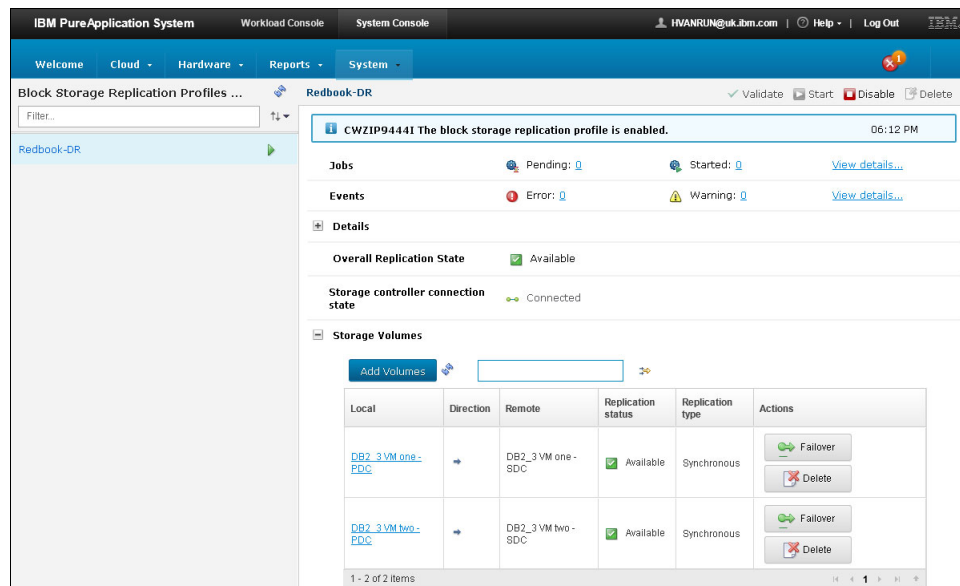


Figure 5-75 Block Storage Volumes with Replication status, Available

5.6.5 Validate the Virtual System Instance on the active system

See 5.7, “Validation” on page 160 for how to validate that the Virtual System Instance on the active system is working as expected. Using a DB2 client, you again create a table and insert a record. Example 5-12 shows how to commit these transactions. Creating a table and inserting a record is a simple but effective mechanism to store data in the DB2 database.

Note: After these transactions are committed, the synchronous DB2 HADR standby database in the PDC will have written those transactions to disk as well. Because you also have synchronous block storage replication setup, that data has also been committed to the Block Storage Volumes in the SDC.

Example 5-12 Committing a simple transaction from a DB2 client

```
db2 "CREATE TABLE AUTHOR (AUTHOR_NUMBER INT NOT NULL PRIMARY KEY, AUTHOR_NAME
VARCHAR(20) NOT NULL)"
db2 "INSERT INTO AUTHOR (AUTHOR_NUMBER, AUTHOR_NAME) VALUES (1, 'Margaret
Ticknor')"
[db2inst1@ipas-lpar-9-3-171-25 ~]$ db2 "select * from author"
```

```
AUTHOR_NUMBER  AUTHOR_NAME
-----
```

```
1 Margaret Ticknor
```

```
1 record(s) selected.
```

```
[db2inst1@ipas-lpar-9-3-171-25 ~]$ db2 connect reset
DB20000I The SQL command completed successfully.
```

5.6.6 Planned Failover to SDC

At this point, you are going to simulate a *planned* failover to the SDC. This option is useful when planned downtime is required, perhaps for the entire PDC or for the PureApplication System in the PDC.

Note: An *unplanned* failover is described in 5.6.7, “Unplanned Failover to SDC” on page 156.

Stop DB2 on the Virtual System Instance in the PDC

Note: When you are just performing a failover *test*, you can skip this step and leave DB2 running in the PDC. This avoids an outage of DB2 in the PDC altogether.

See “Stop DB2 on the Virtual System Instance in the SDC” on page 134 and follow the instructions to stop the DB2 instance on *both* VMs of the Virtual System Instance in the PDC. Example 5-13 provides a summary of the process.

Example 5-13 Stopping DB2 on the Virtual System Instance in the PDC

```
-bash-4.1# chrg -o Offline db2_db2inst1_ipas-lpar-9-3-171-24_0-rg
-bash-4.1# chrg -o Offline db2_db2inst1_ipas-lpar-9-3-171-25_0-rg
-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Request=Lock Nominal=Online
```

```

'- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
Control=SuspendedPropagated
  |- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ipas-lpar-9-3-171-24
  '- Online
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ipas-lpar-9-3-171-25
Offline IBM.ResourceGroup:db2_db2inst1_ipas-lpar-9-3-171-24_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ipas-lpar-9-3-171-24_0-rs
  '- Offline
IBM.Application:db2_db2inst1_ipas-lpar-9-3-171-24_0-rs:ipas-lpar-9-3-171-24
Offline IBM.ResourceGroup:db2_db2inst1_ipas-lpar-9-3-171-25_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ipas-lpar-9-3-171-25_0-rs
  '- Offline
IBM.Application:db2_db2inst1_ipas-lpar-9-3-171-25_0-rs:ipas-lpar-9-3-171-25
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
  |- Online IBM.PeerNode:ipas-lpar-9-3-171-25:ipas-lpar-9-3-171-25
  '- Online IBM.PeerNode:ipas-lpar-9-3-171-24:ipas-lpar-9-3-171-24
Online IBM.Equivalency:db2_db2inst1_ipas-lpar-9-3-171-24_0-rg_group-equ
  '- Online IBM.PeerNode:ipas-lpar-9-3-171-24:ipas-lpar-9-3-171-24
Online IBM.Equivalency:db2_db2inst1_ipas-lpar-9-3-171-25_0-rg_group-equ
  '- Online IBM.PeerNode:ipas-lpar-9-3-171-25:ipas-lpar-9-3-171-25
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth1:ipas-lpar-9-3-171-24
  '- Online IBM.NetworkInterface:eth1:ipas-lpar-9-3-171-25

```

Detach Block Storage volumes from the VMs in the PDC

Note: When just performing a failover test, you can skip this step and leave DB2 running in the PDC. This avoids an outage of DB2 in the PDC altogether.

With DB2 no longer running on the Virtual System Instance in the PDC, you can safely detach the Block Storage Volumes from the VMs. See “Detach Block Storage volumes from the VMs in SDC” on page 136 for more details about this process.

Note: Always make sure that no open files exist on the mounted file system for the Block Storage Volumes *before* detaching them.

Clone Block Storage Volume Group in the SDC

Now you clone the block storage volume group in the second data center so you can attach those to the VMs and start DB2 again. The advantage of this process is that it allows for a planned failover and a planned failover test.

Note: Remember that the VMs of the Virtual System Instance in the SDC are still running (required for attach and detach operations), but that DB2 is not running (as the file system for the DB2 data is not mounted).

To clone the Block Storage Volume, use the following steps:

1. From the System Console, select **Cloud** → **Storage Volumes Groups** and select the Storage Volume Group, **DB2_3** (see Figure 5-76).

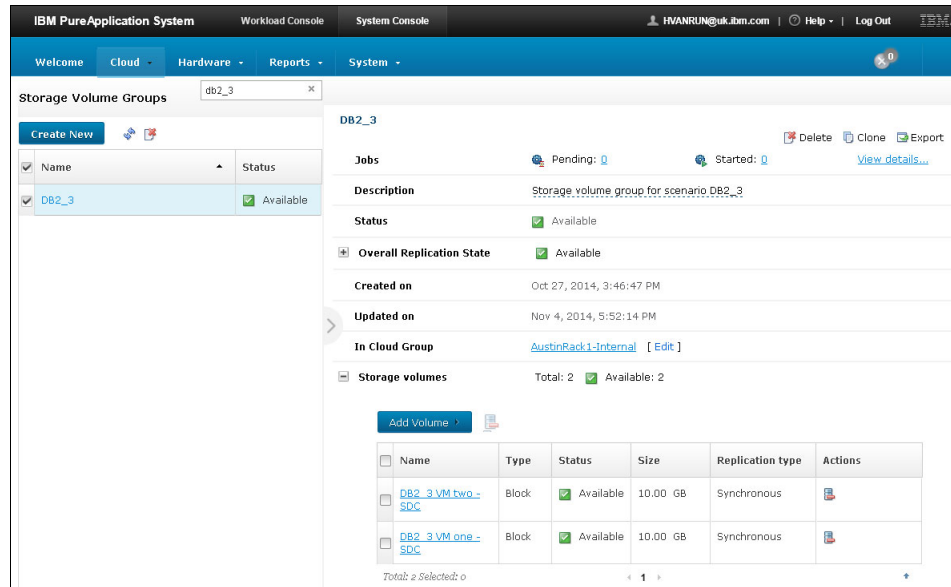


Figure 5-76 Selecting the Storage Volume for cloning process

2. Click **Clone**, provide the following details, and then click **OK** to perform the clone operation (see Figure 5-77):
 - Name: DB2_3_clone_planned_failover
 - Description: Clone for planned failover from PDC to SDC
 - Storage volume naming configuration:
 - New name for DB2_3 VM two - SDC: DB2_3 VM two - SDC clone
 - New name for DB2_3 VM one - SDC: DB2_3 VM one - SDC clone

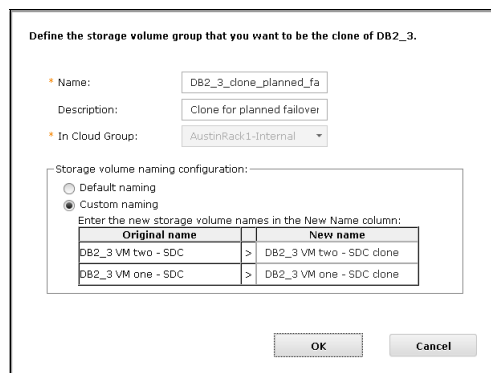


Figure 5-77 Configure the clone

- After some time, the clone operations complete. Examine the newly created Storage Volume Group, DB2_3_clone_planned_failover (see Figure 5-78).

Note: The cloned Block Storage Volumes appear with *Asynchronous* as the replication type. This is the default for Block Storage Volumes that do not take part in replication.

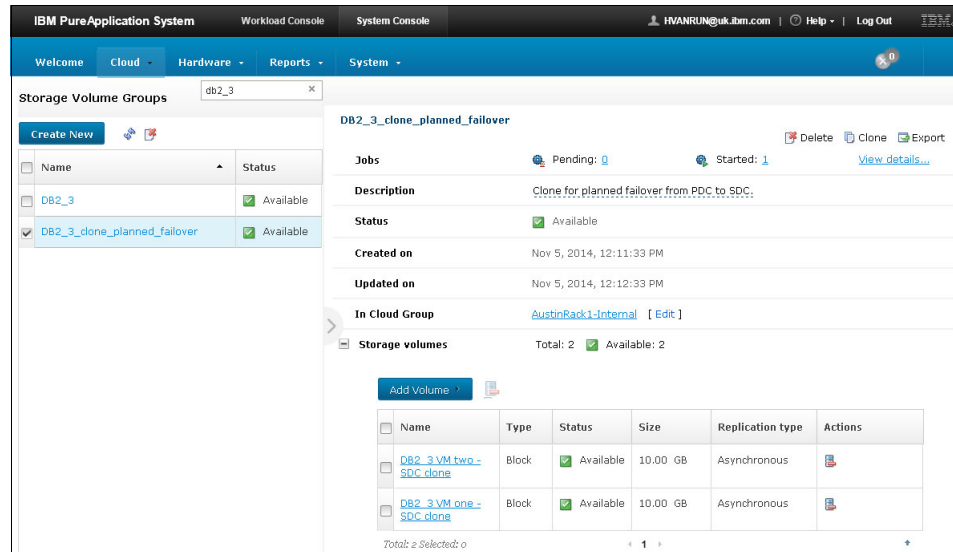


Figure 5-78 Examining the clone

Setup Block Storage Replication from the cloned volumes

Now, you can set up Block Storage Replication for the cloned Block Storage Volumes.

Start by stopping the replication of the original Block Storage Volumes that were configured for synchronous replication in “Configure Block Storage replication” on page 137 using these steps:

- Log on to the System Console of the system in the PDC and select **System** → **Block Storage Replication**.
- Select the Block Storage Replication Profile **Redbooks-DR** and find the Block Storage replication definitions that are listed in Table 5-10.

Table 5-10 Block Storage replication definitions to be deleted

| Block Storage Replication Profile | Source Block Storage Volume (PDC) | Target Block Storage Volume (SDC) |
|-----------------------------------|-----------------------------------|-----------------------------------|
| Redbooks DR | DB2_3 VM One - PDC | DB2_3 VM One - SDC |
| Redbooks DR | DB2_3 VM Two - PDC | DB2_3 VM One - SDC |

3. Delete each of the Block Storage replication deviations by clicking the corresponding **Delete** button (see Figure 5-79).

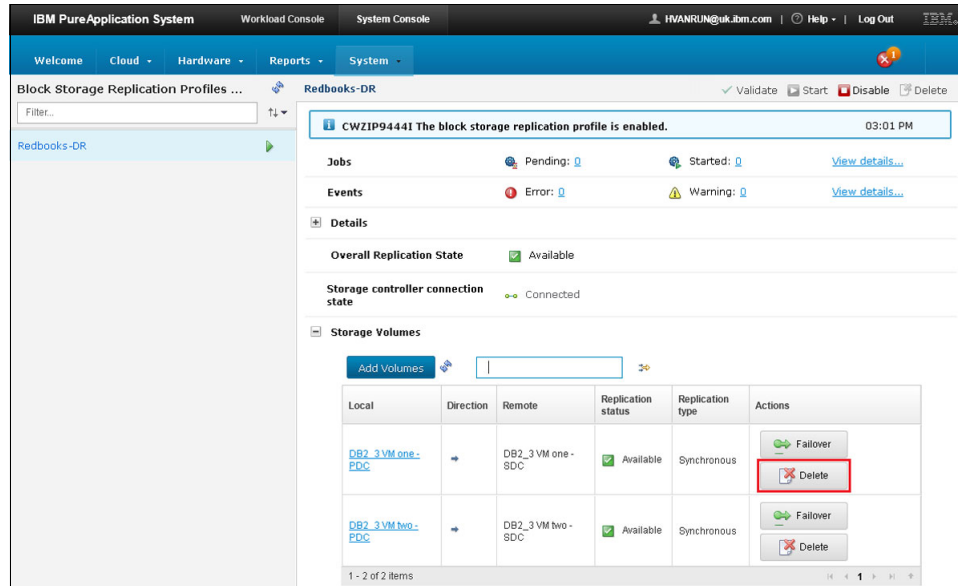


Figure 5-79 Deleting Block Storage Volumes

4. Click **OK** to confirm you want to stop replicating the corresponding pair of Block Storage Volumes (see Figure 5-80).



Figure 5-80 Confirmation notice

5. Repeat steps 2 on page 145 through 4 to delete the second Block Storage Volume.
6. Now, log on to the System Console of the system in the SDC. Select **System** → **Block Storage Replication** to confirm that the target Block Storage Volumes are no longer listed (see Figure 5-81).

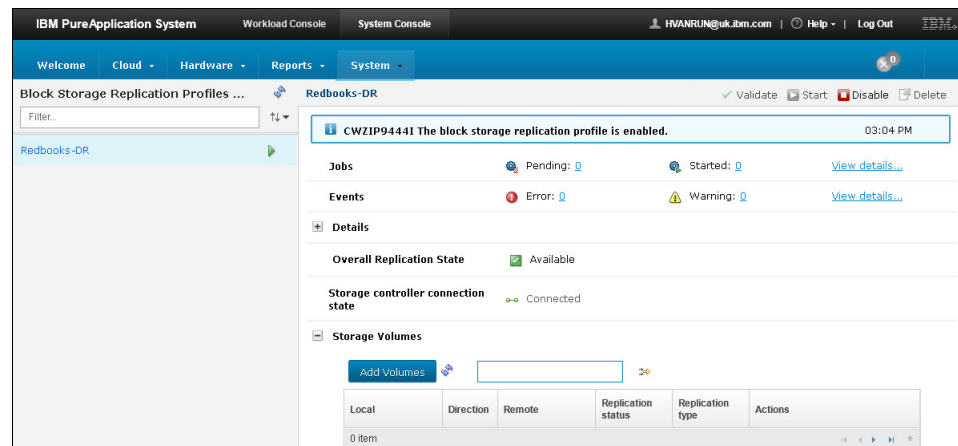


Figure 5-81 Verify no listings for deleted Block Storage Volumes

- Next, configure the Replication of the cloned Block Storage Volumes from the SDC to the PDC. Follow the steps in “Configure Block Storage replication” on page 137 to configure Block Storage Replication as listed in Table 5-11.

Table 5-11 New configuration for Block Storage Replication

| Block Storage Replication Profile | Source Block Storage Volume (SDC) | Target Block Storage Volume (PDC) |
|-----------------------------------|-----------------------------------|-----------------------------------|
| Redbooks DR | DB2 VM one - SDC clone | DB2 VM one - PDC |
| Redbooks DR | DB2 VM two - SDC clone | DB2 VM one - PDC |

After completing the steps 1 on page 145 through 7, the Block Storage Replication Profile on the system in the SDC should look like Figure 5-82.

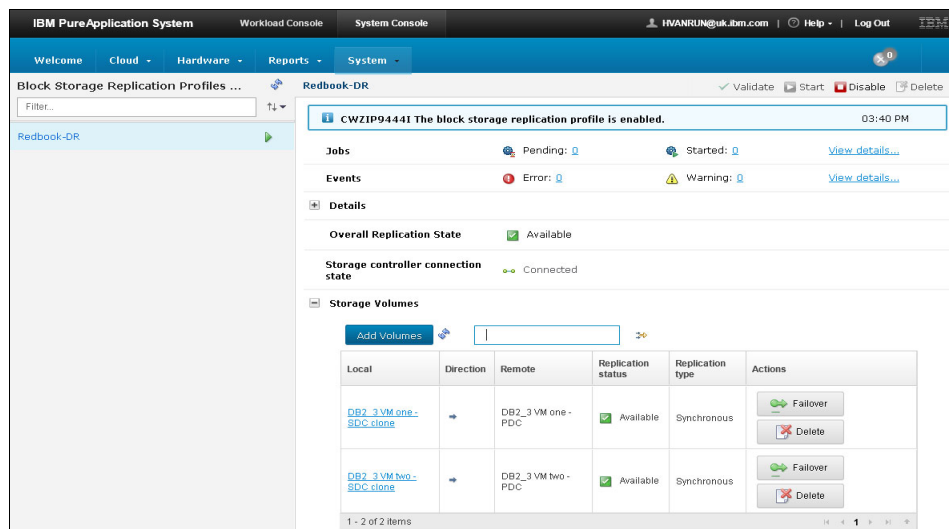


Figure 5-82 New Block Storage Replication

Attach cloned Block Storage Volumes to Virtual System Instance in SDC

Now attach the cloned Block Storage Volumes to the Virtual System Instance in the SDC. For both VMs, use the following steps to attach their corresponding Block Storage Volume:

- Log on to the VM and confirm that no file system is mounted under /db2data (see Example 5-14).

Example 5-14 Confirm that no file system is mounted under /db2data

```
[db2inst1@ausipas089 ~]$ mount
/dev/mapper/vg_root-LogVol100 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/group1-LV9067 on /myinst type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

- From the Virtual System Instance in the Workload Console, expand the Virtual Machines section on the right side of the window (see Figure 5-83).

| Name | Public IP | VM Status | CPU | Memory | Action |
|------------------------------|------------|-----------|-----|--------|--------|
| DB2_VM_one 11415087995171 | 9.3.169.88 | Running | 1% | 7% | Manage |
| DB2_VM_two 11415087995170 | 9.3.169.89 | Running | 1% | 8% | Manage |

Figure 5-83 Virtual machine perspective

- Expand the appropriate VM, and look for the *Block storage* section. In that section, find the Block Storage entry for the mount point, /db2data, for an *ext3* file system. Select the Block Storage Volumes, **DB2_3 VM one - SDC clone**, and click **Attach** to attach the Block Storage Volume. See Figure 5-84.

| Mount Point | File Type | Volume | Status | Action |
|-------------|-----------|--------------------------|--------|---------------|
| /db2data | ext3 | DB2_3 VM one - SDC clone | | Attach Create |

Note: the virtual machine needs to be running in order to perform attach and detach operations.

Figure 5-84 Attaching the Block Storage Volume

- After attaching the block storage, a file system is mounted on /db2data. Confirm this by running the `mount` command on the VM (see Example 5-15).

Example 5-15 Confirm that the file system of the Block Storage Volume has been mounted

```
[db2inst1@ausipas089 ~]$ mount
/dev/mapper/vg_root-LogVol100 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/group1-LV9067 on /myinst type ext3 (rw)
/dev/sdc1 on /db2data type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

Start DB2 on the Virtual System Instance in the SDC

You next proceed to start DB2 on the Virtual System Instance in the SDC. Start by running the `lssam` command, as shown in Example 5-16. You want to obtain the Resource Groups of the Virtual System Instance by using the `lssam` command. Use the output to populate Table 5-12 on page 149.

Example 5-16 Obtaining the Resource Groups of the Virtual System Instance in the SDC

```
[db2inst1@ausipas088 ~]$ lssam
Offline IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ausipas088_0-rs
    '- Offline IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Offline IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ausipas089_0-rs
    '- Offline IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Request=Lock
Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
Control=StartInhibitedBecauseSuspended
```



```

|- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
  |- Online IBM.PeerNode:ausipas088:ausipas088
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth1:ausipas089
  '- Online IBM.NetworkInterface:eth1:ausipas088

```

Table 5-12 Overview of the three Resource Groups of the Virtual System Instance in the SDC

| Resource group | Description | Expected status |
|-------------------------------|---|-----------------|
| db2_db2inst1_ausipas088_0-rg | DB2 instance “db2inst1” on host “ausipas088” | Online |
| db2_db2inst1_ausipas089_0-rg | DB2 instance “db2inst1” on host “ausipas089” | Online |
| db2_db2inst1_db2inst1_ITS0-rg | Controls which database is the primary and standby. | Online |

To start DB2 on the Virtual System Instance in the SDC, perform the following steps:

1. Log on to the VM corresponding to the first VM, ausipas088. Run the command **chrg**, as shown in Example 5-17. Running that command starts the Tivoli System Automation Resource Group corresponding to the DB2 instance on ausipas088 from Table 5-12.

Note: To start the Tivoli System Automation Resource Group, you must run the **chrg** command as *root*.

Example 5-17 Starting the Resource Group corresponding to the DB2 instance on the first VM

```
-bash-4.1# chrg -o Online db2_db2inst1_ausipas088_0-rg
```

2. Run the **lssam** command. Notice that the Resource Group for the DB2 instance on host ausipas088 shifts to online soon after issuing that command, as shown in Example 5-18.

Example 5-18 Confirming that the Resource Group for the DB2 instance is now online.

```

-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
    '- Online
IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Offline IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Offline
  '- Offline IBM.Application:db2_db2inst1_ausipas089_0-rs
    '- Offline
IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Request=Lock
Nominal=Online

```

```

'- Offline IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
Control=StartInhibitedBecauseSuspended
  |- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
  |- Online IBM.PeerNode:ausipas088:ausipas088
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth1:ausipas089
  '- Online IBM.NetworkInterface:eth1:ausipas088

```

- Now, log on as db2inst1 to host ausipas088 to determine the status of the local DB2 instance. Run the **db2pd** command to confirm the state of the DB2 instance. Example 5-19 shows that the **db2pd** command is unable to attach to the database manager, which implies that the local instance is not running.

Example 5-19 Confirming that the DB2 instance on the standby host is no longer running

```

[[db2inst1@ausipas088 ~]$ db2pd -

Database Member 0 -- Active -- Up 0 days 00:01:30 -- Date
2014-11-05-14.59.37.743558

```

- Repeat steps 1 on page 149 through 3 to start the DB2 instance on the second database VM, ausipas089. After completing those steps, the DB2 instance should be running on *both* VMs. The output from the **lssam** command confirms this (as shown in Example 5-20).

Example 5-20 Output from lssam command when both DB2 instances are started

```

-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
    '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Online IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs
    '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Request=Lock
Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
Control=StartInhibitedBecauseSuspended
  |- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
  '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
  |- Online IBM.PeerNode:ausipas088:ausipas088
  '- Online IBM.PeerNode:ausipas089:ausipas089

```

```

Online IBM.Equivalency:db2_public_network_0
|- Online IBM.NetworkInterface:eth1:ausipas089
'- Online IBM.NetworkInterface:eth1:ausipas088

```

Manually updating the DB2 HADR configuration on the VMs

You now need to make some manual changes to the DB2 HADR configuration of the database, ITSO.

The configuration of the database ITSO is stored in /db2data, which is a file system on a Block Storage Volume. Run the Virtual System Instance in the SDC with the data from the Block Storage Volumes of the Virtual System Instance in the PDC. As a result, the DB2 HADR configuration contains references to host names from the Virtual System Instance in the PDC.

The host names for the VMs of the Virtual System Instances in the PDC and SDC are listed in Table 5-13.

Note: The short names are listed for convenience. All hosts are part of the domain .test.ibm.com.

Table 5-13 Overview of host names for the Virtual System Instances in the PDC and the SDC

| Virtual System Instance | First VM | Second VM |
|-------------------------|----------------------|------------|
| PDC | ipas-lpar-9-3-171-25 | ausipas088 |
| SDC | ipas-lpar-9-3-171-24 | ausipas089 |

To manually update the DB2 HADR configuration on the VMs of the Virtual System Instance in the SDC, perform the following steps:

1. Example 5-21 shows the current database configuration parameters, HADR_LOCAL_HOST and HADR_REMOTE_HOST, of the ITSO database on the first VM.

Note: These parameters still use the host names of the VMs belonging to the Virtual System Instance in the PDC as previously stated.

Example 5-21 Extracting HADR configuration parameters of the ITSO database on the first VM

```

[db2inst1@ausipas088 ~]$ db2 get db cfg for itso | grep "host name"
HADR local host name           (HADR_LOCAL_HOST) = ipas-lpar-9-3-171-25
HADR remote host name         (HADR_REMOTE_HOST) = ipas-lpar-9-3-171-24

```

2. You need to update these two DB2 configuration parameters of the ITSO database on the first VM. Example 5-22 shows how to update these parameters and modify them to correspond with the host names of the Virtual System Instance in the SDC (as listed previously in Table 5-13).

Note: Set HADR_LOCAL_HOST to the host name of the local server, ausipas088. Set HADR_REMOTE_HOST to the host name of the other VM, ausipas089.

Example 5-22 Manually changing the DB2 HADR host names on first VM

```

[db2inst1@ausipas088 ~]$ db2 update db cfg for itso using HADR_LOCAL_HOST
ausipas088

```

```
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
[db2inst1@ausipas088 ~]$ db2 update db cfg for itso using HADR_REMOTE_HOST
ausipas089
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
```

```
[[db2inst1@ausipas088 ~]$ db2 get db cfg for itso | grep "host name"
HADR local host name (HADR_LOCAL_HOST) = ausipas088
HADR remote host name (HADR_REMOTE_HOST) = ausipas089
```

- Repeat step 2 on page 151 for the second VM, as shown in Example 5-23.

Note: This time set HADR_LOCAL_HOST to ausipas089 and HADR_REMOTE_HOST to ausipas088.

Example 5-23 Manually changing the DB2 HADR host names on second VM

```
[db2inst1@ausipas089 ~]$ db2 update db cfg for itso using HADR_LOCAL_HOST
ausipas089
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, the database
must be shutdown and reactivated before the configuration parameter changes
become effective.
[db2inst1@ausipas089 ~]$ db2 update db cfg for itso using HADR_REMOTE_HOST
ausipas088
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, the database
must be shutdown and reactivated before the configuration parameter changes
become effective.
```

- The correct DB2 HADR configuration is in place. However, the status of the `lssam` command still shows the Resource Group, `db2_db2inst1_db2inst1_ITSO-rg`, as *pending online*, as shown in Example 5-24. To address this, you need to restart the DB2 instance on both VMs.

Example 5-24 The Resource Group, db2_db2inst1_ITSO-rg, still shows as pending online

```
-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
        '- Online
IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Online IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs
        '- Online
IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITSO-rg Request=Lock
Nominal=Online
    '- Offline IBM.Application:db2_db2inst1_db2inst1_ITSO-rs
Control=StartInhibitedBecauseSuspended
    '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITSO-rs:ausipas088
    '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITSO-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
```

```

    '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
    |- Online IBM.PeerNode:ausipas088:ausipas088
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth1:ausipas089

```

5. Stop the Resource Groups corresponding to the DB2 instances on both VMs, as shown in Example 5-25.

Example 5-25 Commands to stop the Resource Groups for both DB2 instances

```

-bash-4.1# chrg -o Offline db2_db2inst1_ausipas089_0-rg
-bash-4.1# chrg -o Offline db2_db2inst1_ausipas088_0-rg

```

6. Wait a few seconds to make sure that both DB2 instances have been stopped. Use the `lssam` command to confirm that the Resources Groups corresponding to the DB2 instances are **Offline** as shown in Example 5-26.

Example 5-26 The Resource Groups for the DB2 instances are now Offline

```

-bash-4.1# lssam
Offline IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Offline
    '- Offline IBM.Application:db2_db2inst1_ausipas088_0-rs
        '- Offline
IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Offline IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Offline
    '- Offline IBM.Application:db2_db2inst1_ausipas089_0-rs
        '- Offline
IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Request=Lock
Nominal=Online
    '- Offline IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
Control=StartInhibitedBecauseSuspended
    |- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
    '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
    |- Online IBM.PeerNode:ausipas088:ausipas088
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth1:ausipas089
    '- Online IBM.NetworkInterface:eth1:ausipas088

```

7. Start both DB2 instances again using the `chrg` command, as shown in Example 5-27.

Example 5-27 Commands to start the Resource Groups for both DB2 instances

```

-bash-4.1# chrg -o Online db2_db2inst1_ausipas088_0-rg
-bash-4.1# chrg -o Online db2_db2inst1_ausipas089_0-rg

```

- Use the `lssam` command to confirm that all three Resource Groups are online, as shown in Example 5-28.

Example 5-28 All three Resource Groups are now Online

```
-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
        '- Online
IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Online IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs
        '- Online
IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
        |- Online
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
    '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-equ
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-equ
    |- Online IBM.PeerNode:ausipas088:ausipas088
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth1:ausipas089
    '- Online IBM.NetworkInterface:eth1:ausipas088
```

- While logged on as `db2inst1` on the first VM, run the command, `db2pd -hadr -db itso`. Example 5-29 shows the output of this command, which indicates that the ITS0 database is started as the primary.

Example 5-29 Confirming that the database, ITS0, is started on the first VM as the primary

```
[db2inst1@ausipas088 ~]$ db2pd -hadr -db itso
```

```
Database Member 0 -- Database ITS0 -- Active -- Up 0 days 00:34:19 -- Date
2014-11-05-15.44.26.172751
```

```

      HADR_ROLE = PRIMARY
      REPLAY_TYPE = PHYSICAL
      HADR_SYNCMODE = SYNC
      STANDBY_ID = 1
      LOG_STREAM_ID = 0
      HADR_STATE = PEER
      HADR_FLAGS =
      PRIMARY_MEMBER_HOST = ausipas088
      PRIMARY_INSTANCE = db2inst1
      PRIMARY_MEMBER = 0
      STANDBY_MEMBER_HOST = ausipas089
      STANDBY_INSTANCE = db2inst1
      STANDBY_MEMBER = 0
      HADR_CONNECT_STATUS = CONNECTED
      HADR_CONNECT_STATUS_TIME = 11/05/2014 15:10:21.174085 (1415200221)
```

```

HEARTBEAT_INTERVAL(seconds) = 30
HADR_TIMEOUT(seconds) = 150
TIME_SINCE_LAST_RECV(seconds) = 5
PEER_WAIT_LIMIT(seconds) = 0
LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.003363
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.219
LOG_HADR_WAIT_COUNT = 65
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 19800
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
PRIMARY_LOG_FILE,PAGE,POS = S0000045.LOG, 10, 232800446
STANDBY_LOG_FILE,PAGE,POS = S0000045.LOG, 10, 232800446
HADR_LOG_GAP(bytes) = 0
STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000045.LOG, 10, 232800446
STANDBY_RECV_REPLAY_GAP(bytes) = 0
PRIMARY_LOG_TIME = 11/05/2014 15:25:31.000000 (1415201131)
STANDBY_LOG_TIME = 11/05/2014 15:25:31.000000 (1415201131)
STANDBY_REPLAY_LOG_TIME = 11/05/2014 15:25:31.000000 (1415201131)
STANDBY_RECV_BUF_SIZE(pages) = 4298
STANDBY_RECV_BUF_PERCENT = 0
STANDBY_SPOOL_LIMIT(pages) = 25600
STANDBY_SPOOL_PERCENT = 0
PEER_WINDOW(seconds) = 120
PEER_WINDOW_END = 11/05/2014 15:46:21.000000 (1415202381)
READS_ON_STANDBY_ENABLED = N

```

Confirm that the database is available again

As noted previously in “Manually updating the DB2 HADR configuration on the VMs” on page 151, the host names of the VMs where DB2 is now running have changed. For DB2 clients to connect to DB2 in the SDC, some changes to the DB2 client configuration need to be made.

In a more realistic implementation, the use of an external load balancer or DNS aliases to access the VMs running VM is more appropriate. DB2 clients are always configured using the load balancer or DNS aliases. In a failover to the other data center, there is no need to reconfigure the DB2 clients.

To confirm that the DB2 database ITSO is available again, complete the following steps:

1. Log on as db2inst1 to the DB2 client and run the **db2 uncatalog database itso** command to uncatalog the database ITSO. Then run the **db2 uncatalog node itsohadr** command to uncatalog the node ITSOHADR. The sample output for both commands is shown in Example 5-30.

Example 5-30 Uncatalog the database, ITSO, and the node, ITSOHADR

```

[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 uncatalog database itso
DB20000I The UNCATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 uncatalog node itsohadr
DB20000I The UNCATALOG NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.

```

2. Now, add a node and database to the catalog, as shown in Example 5-31.

Note: Use the FQDN of the first VM in the SDC, which is the VM that is hosting the primary ITSO database.

Example 5-31 Catalog the database, ITSO, and the updated node, ITSOHADR

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 catalog tcpip node itsohadr remote
ausipas088.austin.ibm.com server 50000
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 catalog database itso at node itsohadr
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
```

3. With the updated catalog entries, you can now connect to the DB2 database, ITSO, in the SDC. Example 5-32 shows how to perform a simple query, which confirms that you successfully performed a failover to the SDC.

Example 5-32 Performing a simple query to confirm that the data is still in the table

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 connect to itso user itsouser
Enter current password for itsouser:
```

Database Connection Information

```
Database server      = DB2/LINUX8664 10.5.3
SQL authorization ID = ITSOUSER
Local database alias = ITSO
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select * from author"
```

```
AUTHOR_NUMBER  AUTHOR_NAME
-----
1 Margaret Ticknor
```

```
1 record(s) selected.
```

5.6.7 Unplanned Failover to SDC

An *unplanned* failover to the SDC is the appropriate action if the PDC unexpectedly fails. Examples of causes that can lead to an unplanned outage include natural disasters or an unplanned power outage.

Note: With an unplanned failover, there is always going to be a brief outage of DB2. Take into account the time to detect that the PDC is suffering from an outage, plus the time it takes to failover to the SDC. This gives an indication for the *Recovery Time Objective*. Because this scenario is using synchronous Block Storage Replication, there is no loss of any committed transactions. Hence, the *Recovery Point Objective* is 0.

Failover of Block Storage Volumes Replication

With an unplanned outage of the PDC, this scenario assumes that there is no longer access to the PureApplication System (or any of the VMs) in the PDC. As a result, the only option available to fail over the Block Storage Volumes is from the System Console of the system in the SDC.

The Block Storage Volumes that need to fail over are noted in the following list:

- ▶ DB2_3 VM One - SDC
- ▶ DB2_3 VM Two - SDC

To perform the failover of these Block Storage Volumes, use the following steps:

1. Log on to System Console in the SDC and select **System** → **Block Storage Replication** and select the Block Storage Replication Profile, **Redbooks-DR** (see Figure 5-85).

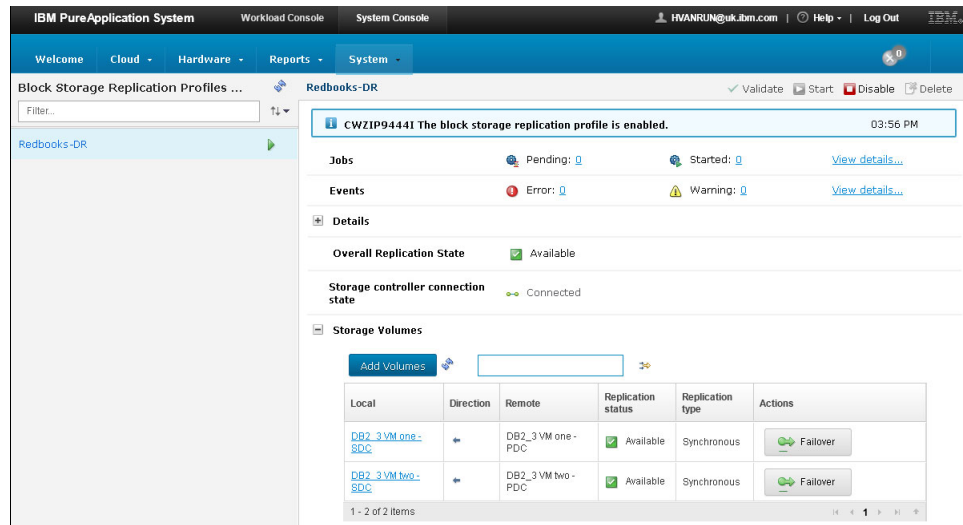


Figure 5-85 Select the appropriate Block Storage Replication Profile

2. For each of the Block Storage Volumes, use the following process to do perform the actual Failover:
 - a. Under Storage Volumes, click **Failover** (see Figure 5-86).

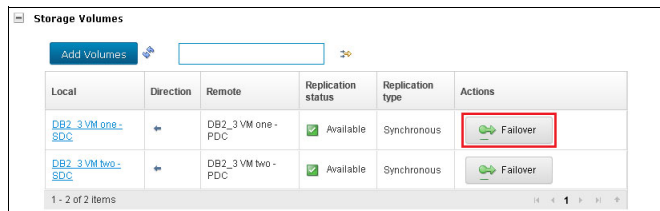


Figure 5-86 Engaging Failover

- b. Click **OK** to perform the unplanned failover of the Block Storage Volume (see Figure 5-87).

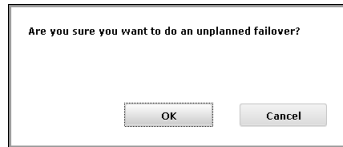


Figure 5-87 Failover engaged

3. After both Block Storage Volumes have been failed over, they no longer appear under Storage Volumes of the Block Storage Replication Profile (see Figure 5-88).

Note: The two Block Storage Volumes are no longer part of a replication pair. They have become ordinary Block Storage Volumes. As a result, they can be attached to VMs without any further steps.

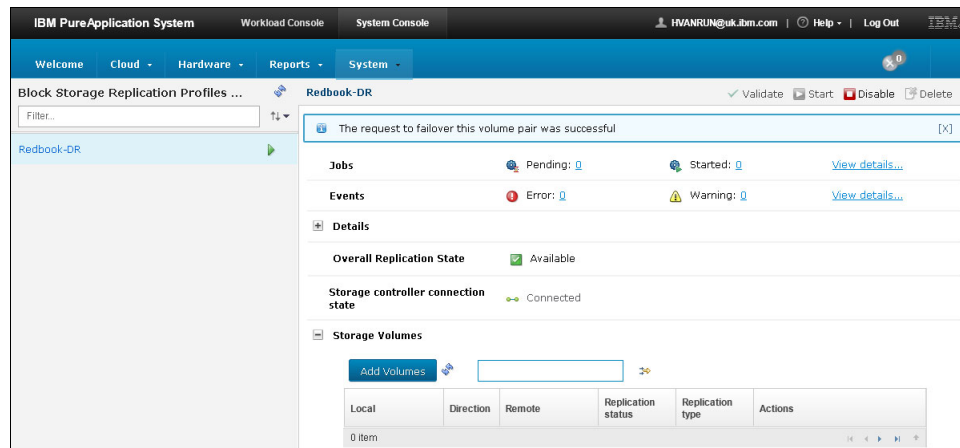


Figure 5-88 Confirm the change in the Storage Volumes Profile

Attach Block Storage Volumes to Virtual System Instance in SDC

You can now attach the Block Storage Volumes to the Virtual System Instance in the SDC. See “Attach cloned Block Storage Volumes to Virtual System Instance in SDC” on page 147.

Note: This time, you attach the actual Block Storage Volumes in the SDC because you did not clone them.

Start DB2 on Virtual System Instance in SDC

See “Start DB2 on the Virtual System Instance in the SDC” on page 148 for details about how to start DB2 on the Virtual System Instance in the SDC.

Manually updating the DB2 HADR configuration on VMs

You now need to make some manual changes to the DB2 HADR configuration of the database, ITSO. See “Manually updating the DB2 HADR configuration on the VMs” on page 151 for more details about how to do this and its importance.

After DB2 is running on both VMs, the output from the `lssam` command should look as shown Example 5-33.

Example 5-33 All three Resource Groups are Online

```

-bash-4.1# lssam
Online IBM.ResourceGroup:db2_db2inst1_ausipas088_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs
        '- Online IBM.Application:db2_db2inst1_ausipas088_0-rs:ausipas088
Online IBM.ResourceGroup:db2_db2inst1_ausipas089_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs
        '- Online IBM.Application:db2_db2inst1_ausipas089_0-rs:ausipas089
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_ITS0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs
        |- Online IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas088
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_ITS0-rs:ausipas089
Online IBM.Equivalency:db2_db2inst1_ausipas088_0-rg_group-eq
    '- Online IBM.PeerNode:ausipas088:ausipas088
Online IBM.Equivalency:db2_db2inst1_ausipas089_0-rg_group-eq
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_db2inst1_db2inst1_ITS0-rg_group-eq
    |- Online IBM.PeerNode:ausipas088:ausipas088
    '- Online IBM.PeerNode:ausipas089:ausipas089
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth1:ausipas089
    '- Online IBM.NetworkInterface:eth1:ausipas088

```

Confirm that the database is available again

See “Confirm that the database is available again” on page 155 for more details. This completes the unplanned failover to the SDC.

Note: When the PDC has been recovered, set up Block Storage Replication again.

Setup Block Storage Replication after the PDC has been recovered

This section does not document the steps in detail (because they vary depending on what was lost in the PDC). After there is a deployed Virtual System Instance up and running in the PDC again, take the following actions:

1. Stop DB2 on both VMs, as documented in “Stop DB2 on the Virtual System Instance in the SDC” on page 134.
2. Detach the Block Storage Volumes from both VMs, as documented in “Detach Block Storage volumes from the VMs in SDC” on page 136.
3. Configure Block Storage Replication from the SDC to the PDC, as documented in Figure 5-78 on page 145. Table 5-14 shows how the setup should appear in this case.

Table 5-14 Configuration for Block Storage Replication after the PDC has been recovered

| Block Storage Replication Profile | Source Block Storage Volume (SDC) | Target Block Storage Volume (PDC) |
|-----------------------------------|-----------------------------------|-----------------------------------|
| Redbooks DR | DB2 VM one - SDC | DB2 VM one - PDC |
| Redbooks DR | DB2 VM two - SDC | DB2 VM one - PDC |

Note: After Block Storage Replication has been setup, it also allows for a failback to the PDC if required.

5.7 Validation

One of the key benefits of IBM PureApplication System is that the built-in expertise of patterns helps accelerate the implementation of complex middleware solutions. However, it is always a good idea to perform a number of rudimentary tests to validate that the deployed pattern indeed delivers the expected result.

In the case of this scenario's DB2 Virtual System Pattern, the following items are confirmed:

- ▶ Configure the DB2 client
- ▶ Connect to the DB2 database, using itsouser credentials and perform a simple query
- ▶ Confirm the DB2 HADR roles of both the primary and standby DB2 servers

5.7.1 Configure the DB2 client

First, you need to carefully document the host names and IP addresses of the two Virtual Machines that have been deployed across the two PureApplication Systems. As shown in Figure 5-89, you can obtain the (public) IP addresses by expanding the **Virtual machine perspective** of the deployed Virtual System Instance.

| Name | Public IP | VM Status | CPU | Memory | Action |
|-------------------------------|-------------|-----------|-----|--------|--------|
| DB2_VM_one. 11413970235089 | 9.3.171.148 | Running | 4% | 9% | Manage |
| DB2_VM_two. 11413970235088 | 9.3.169.66 | Running | 5% | 11% | Manage |

Figure 5-89 Obtaining IP addresses of the Virtual Machines of the deployed Virtual System Instance

By expanding each of the Virtual Machines, you can also find the host name. For convenience, Table 5-15 is populated with the Virtual Machine names, IP addresses, and host names.

Table 5-15 Virtual Machine names, IP addresses, and host names

| Virtual Machine Name | IP address | Host name |
|---------------------------|-------------|--------------------------------------|
| DB2_VM_one.11413970235089 | 9.3.171.148 | ipas-lpar-9-3-171-148.austin.ibm.com |
| DB2_VM_two.11413970235088 | 9.3.169.66 | ausipas066.austin.ibm.com |

You can now configure the DB2 client described in 5.2, “DB2 Client Setup” on page 92. Start by cataloging the node, which is pointed to the first Virtual Machine (DB2_VM_one.1141397023508) as it hosts the primary DB2 database. This is shown in Example 5-34.

Example 5-34 Catalog node hosting primary database on DB2 client

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 catalog tcpip node itsohadr remote
ipas-lpar-9-3-171-148.austin.ibm.com server 50000
```

```
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 list node directory
```

Node Directory

Number of entries in the directory = 1

Node 1 entry:

```
Node name           = ITSOHADR
Comment             =
Directory entry type = LOCAL
Protocol            = TCPIP
Hostname            = ipas-lpar-9-3-171-148.austin.ibm.com
Service name        = 50000
```

With the node cataloged, you can proceed and catalog the database, ITSO, on the DB2 client. Example 5-35 shows how this is done.

Example 5-35 Catalog database on DB2 client

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 catalog database itso at node itsohadr
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 list database directory
```

System Database Directory

Number of entries in the directory = 1

Database 1 entry:

```
Database alias      = ITSO
Database name       = ITSO
Node name           = ITSOHADR
Database release level = 10.00
Comment            =
Directory entry type = Remote
Catalog database partition number = -1
Alternate server hostname =
Alternate server port number =
```

Note: The DB2 client is not aware of the fact that the database, ITSO, has been configured using DB2 HADR. This is reflected by the fact that there is no data populated for *Alternate server host name* and *Alternate server port number*.

5.7.2 Connect to the DB2 database and perform a simple query

Make sure that you can connect to the ITSO catalog database on the DB2 client. This scenario uses the credentials of itsouser. Example 5-36 demonstrates the commands for this process.

Example 5-36 Connecting to the DB2 database, ITSO, from the DB2 client

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 connect to itso user itsouser
Enter current password for itsouser:
```

```
Database Connection Information
```

```
Database server          = DB2/LINUX8664 10.5.3
SQL authorization ID     = ITSOUSER
Local database alias     = ITSO
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select count(*) from sysibm.sysjobs"
```

```
1
-----
0
```

```
1 record(s) selected.
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 connect reset
DB20000I The SQL command completed successfully.
```

After the DB2 client has made its first connection, the client is now aware of the fact that the database, ITSO, is set up with DB2 HADR. You can validate this by reviewing the database alias ITSO on the DB2 client. As shown in Example 5-37, *Alternate server host name* and *Alternate server port number* are automatically populated.

Example 5-37 Confirming that the DB2 client recognizes the DB2 HADR setup of the database, ITSO

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 list database directory
```

```
System Database Directory
```

```
Number of entries in the directory = 1
```

```
Database 1 entry:
```

```
Database alias          = ITSO
Database name           = ITSO
Node name               = ITSOHADR
Database release level  = 10.00
Comment                 =
Directory entry type    = Remote
Catalog database partition number = -1
Alternate server hostname = ausipas066.austin.ibm.com
Alternate server port number = 50000
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$
```

5.7.3 Confirm DB2 HADR roles of primary and standby DB2 servers

You can use the following steps on the primary and standby servers to confirm whether the DB2 database is indeed set up as expected:

1. Log on to the Virtual Machine, DB2_VM_one.11413970235089 server, as the DB2 instance owner (db2inst1) and confirm the HADR status (as shown in Example 5-38). You should see that it reports PRIMARY for HADR_ROLE and CONNECTED for HADR_CONNECT_STATUS.

Example 5-38 Confirming DB2 HADR role of database on primary DB2 server

```
[db2inst1@ipas-lpar-9-3-171-148 ~]$ db2pd -hadr -database itso
```

```
Database Member 0 -- Database ITSO -- Active -- Up 0 days 00:52:07 -- Date
2014-10-22-10.35.10.776418
```

```

                HADR_ROLE = PRIMARY
                REPLAY_TYPE = PHYSICAL
                HADR_SYNCMODE = SYNC
                STANDBY_ID = 1
                LOG_STREAM_ID = 0
                HADR_STATE = PEER
                HADR_FLAGS =
PRIMARY_MEMBER_HOST = ipas-lpar-9-3-171-148
PRIMARY_INSTANCE = db2inst1
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = ausipas066
STANDBY_INSTANCE = db2inst1
STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
                HADR_CONNECT_STATUS_TIME = 10/22/2014 09:43:09.302308 (1413970989)
                HEARTBEAT_INTERVAL(seconds) = 30
                HADR_TIMEOUT(seconds) = 150
                TIME_SINCE_LAST_RECV(seconds) = 1
                PEER_WAIT_LIMIT(seconds) = 0
                LOG_HADR_WAIT_CUR(seconds) = 0.000
                LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.001300
                LOG_HADR_WAIT_ACCUMULATED(seconds) = 2.262
                LOG_HADR_WAIT_COUNT = 2245
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 19800
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
                PRIMARY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                STANDBY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                HADR_LOG_GAP(bytes) = 0
                STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                STANDBY_RECV_REPLAY_GAP(bytes) = 0
                PRIMARY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_REPLAY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_RECV_BUF_SIZE(pages) = 4298
                STANDBY_RECV_BUF_PERCENT = 0
                STANDBY_SPOOL_LIMIT(pages) = 25600
                STANDBY_SPOOL_PERCENT = 0
                PEER_WINDOW(seconds) = 120
                PEER_WINDOW_END = 10/22/2014 10:37:09.000000 (1413974229)
                READS_ON_STANDBY_ENABLED = N
```

- Now, log on to the primary server as the DB2 instance owner (db2inst1) and confirm the HADR status (as shown in Example 5-39). You should see that it reports STANDBY for HADR_ROLE and CONNECTED for HADR_CONNECT_STATUS.

Example 5-39 Confirming DB2 HADR role of database on standby DB2 server

```
[db2inst1@ausipas066 ~]$ db2pd -hadr -database itso
```

```
Database Member 0 -- Database ITSO -- Standby -- Up 0 days 00:54:35 -- Date
2014-10-22-10.37.41.155340
```

```

                HADR_ROLE = STANDBY
                REPLAY_TYPE = PHYSICAL
                HADR_SYNCMODE = SYNC
                STANDBY_ID = 0
                LOG_STREAM_ID = 0
                HADR_STATE = PEER
                HADR_FLAGS =
                PRIMARY_MEMBER_HOST = ipas-lpar-9-3-171-148
                PRIMARY_INSTANCE = db2inst1
                PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = ausipas066
                STANDBY_INSTANCE = db2inst1
                STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
                HADR_CONNECT_STATUS_TIME = 10/22/2014 09:43:09.299742 (1413970989)
                HEARTBEAT_INTERVAL(seconds) = 30
                HADR_TIMEOUT(seconds) = 150
                TIME_SINCE_LAST_RECV(seconds) = 2
                PEER_WAIT_LIMIT(seconds) = 0
                LOG_HADR_WAIT_CUR(seconds) = 0.000
                LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.001300
                LOG_HADR_WAIT_ACCUMULATED(seconds) = 2.262
                LOG_HADR_WAIT_COUNT = 2245
                SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 19800
                SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
                PRIMARY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                STANDBY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                HADR_LOG_GAP(bytes) = 0
                STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000001.LOG, 133, 49651665
                STANDBY_RECV_REPLAY_GAP(bytes) = 0
                PRIMARY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_REPLAY_LOG_TIME = 10/22/2014 10:31:18.000000 (1413973878)
                STANDBY_RECV_BUF_SIZE(pages) = 4298
                STANDBY_RECV_BUF_PERCENT = 0
                STANDBY_SPOOL_LIMIT(pages) = 25600
                STANDBY_SPOOL_PERCENT = 0
                PEER_WINDOW(seconds) = 120
                PEER_WINDOW_END = 10/22/2014 10:39:39.000000 (1413974379)
                READS_ON_STANDBY_ENABLED = N

```

5.8 Testing for outages

When implementing a DB2 HADR solution, an important action step is to demonstrate that the DB2 can handle an outage of either database server. A critical focus is success in the case of primary failure. This section describes a planned and an unplanned outage for the primary database server. It is applicable to all DB2 scenarios described in this chapter.

5.8.1 Planned outage: DB2 takeover

This scenario simulates a *planned* outage of the primary database server. This can occur, for example, when Operating System maintenance needs to be applied, or when the virtual machine needs to be rebooted. This scenario demonstrates that a DB2 client connected to the HADR database is not affected by these examples.

Establish a connection and commit transaction from the DB2 client

This scenario starts by connecting to the database, ITSO, from the DB2 client. Using this connection, create a simple table and insert a single row. This data is *committed* to the primary database, ITSO. Because this database has been set up with *synchronous* HADR, the standby database, ITSO, should have received the committed data before the DB2 client received confirmation that the commit had completed.

Note: Even though this scenario does not make an explicit call to commit the data, when the insert call is performed, it automatically performs a **commit**. The DB2 command line processor (CLP) client uses **autocommit** by default, which is shown in Example 5-40.

Example 5-40 CLP using autocommit by default

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 list command options
```

Command Line Processor Option Settings

```
Backend process wait time (seconds)      (DB2BQTIME) = 1
No. of retries to connect to backend     (DB2BQTRY) = 60
Request queue wait time (seconds)       (DB2RQTIME) = 5
Input queue wait time (seconds)         (DB2IQTIME) = 5
Command options                          (DB2OPTIONS) =
```

| Option | Description | Current Setting |
|--------|---------------------------------------|-----------------|
| -a | Display SQLCA | OFF |
| -b | Auto-Bind | ON |
| -c | Auto-Commit | ON |
| -d | Retrieve and display XML declarations | OFF |
| -e | Display SQLCODE/SQLSTATE | OFF |
| -f | Read from input file | OFF |
| -i | Display XML data with indentation | OFF |
| -j | Return code for system calls | OFF |
| -l | Log commands in history file | OFF |
| -m | Display the number of rows affected | OFF |
| -n | Remove new line character | OFF |
| -o | Display output | ON |
| -p | Display interactive input prompt | ON |
| -q | Preserve whitespaces & linefeeds | OFF |

```

-r    Save output to report file          OFF
-s    Stop execution on command error     OFF
-t    Set statement termination character  OFF
-v    Echo current command                OFF
-w    Display FETCH/SELECT warning messages ON
-x    Suppress printing of column headings OFF
-z    Save all output to output file      OFF

```

Example 5-41 shows the results of committing a transaction to the database ITSO.

Example 5-41 Committing a transaction to the database ITSO

```

[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 connect to itso user itsouser
Enter current password for itsouser:

```

Database Connection Information

```

Database server      = DB2/LINUX8664 10.5.3
SQL authorization ID = ITSOUSER
Local database alias = ITSO

```

```

[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "CREATE TABLE AUTHOR (AUTHOR_NUMBER INT NOT
NULL PRIMARY KEY, AUTHOR_NAME VARCHAR(20) NOT NULL)"
DB20000I The SQL command completed successfully.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "INSERT INTO AUTHOR (AUTHOR_NUMBER,
AUTHOR_NAME) VALUES (1, 'Margaret Ticknor')"
DB20000I The SQL command completed successfully.
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select * from author"

```

```

AUTHOR_NUMBER AUTHOR_NAME
-----

```

```

1 Margaret Ticknor

```

```

1 record(s) selected.

```

Takeover from the DB2 database on the standby server

Before you perform a manual takeover, first confirm that your client is indeed connected to the primary ITSO database. Example 5-42 shows that there is exactly one connection from db2bp. The *Application id* reflects the IP address of the DB2 client.

Example 5-42 Confirming that the DB2 client is indeed connected to the primary ITSO database

```

[db2inst1@ipas-lpar-9-3-171-148 ~]$ db2 list applications

```

```

Auth Id Application  Appl.    Application Id
DB      # of
        Name          Handle
Name    Agents
-----
-----
ITS0USER db2bp          760      9.3.171.4.37669.141022144356
ITS0     1

```

From the DB2 client, you can review the connection details as shown in Example 5-43. The Host name here matches the DB2 server running the primary ITSO database.

Example 5-43 DB2 client connection details confirm that it is connected to the primary ITSO database

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state
```

```
Database Connection State
```

| | |
|----------------------|---|
| Connection state | = Connectable and Connected |
| Connection mode | = SHARE |
| Local database alias | = ITSO |
| Database name | = ITSO |
| Hostname | = ipas-lpar-9-3-171-148.austin.ibm.com |
| Service name | = 50000 |

You can now perform a manual takeover from the DB2 server running the standby ITSO database. Issue the command shown in Example 5-44 to perform the takeover.

Example 5-44 Performing a manual takeover from the DB2 server running the standby ITSO database

```
[db2inst1@ausipas066 ~]$ db2pd -hadr -database itso | grep -E  
"HADR_ROLE|HADR_SYNCMODE|HADR_CONNECT_STATUS "  
          HADR_ROLE = STANDBY  
          HADR_SYNCMODE = SYNC  
          HADR_CONNECT_STATUS = CONNECTED  
[db2inst1@ausipas066 ~]$ db2 takeover hadr on database itso  
[db2inst1@ausipas066 ~]$ db2pd -hadr -database itso | grep -E  
"HADR_ROLE|HADR_SYNCMODE|HADR_CONNECT_STATUS "  
          HADR_ROLE = PRIMARY  
          HADR_SYNCMODE = SYNC  
          HADR_CONNECT_STATUS = CONNECTED
```

Example 5-45 confirms that the DB2 server that was running the primary ITSO database is now hosting the standby.

Example 5-45 Confirming that the DB2 server hosting the old primary for ITSO now hosts the standby

```
[db2inst1@ipas-lpar-9-3-171-148 ~]$ db2pd -hadr -database itso | grep -E  
"HADR_ROLE|HADR_SYNCMODE|HADR_CONNECT_STATUS "  
          HADR_ROLE = STANDBY  
          HADR_SYNCMODE = SYNC  
          HADR_CONNECT_STATUS = CONNECTED
```

Run SQL query from the DB2 client

Now that you are certain that you performed a successful manual takeover, consider again the DB2 client. Before performing the takeover, the client had established a connection with the old primary ITSO database. Example 5-46 shows that the DB2 client still functions as though it is connected to the old primary ITSO database. However, after you perform another query, the DB2 client encounters SQL30108N.

Example 5-46 Query to update DB2 client

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state
```

```
Database Connection State
```

```
Connection state      = Connectable and Connected
Connection mode       = SHARE
Local database alias  = ITSO
Database name         = ITSO
Hostname              = ipas-lpar-9-3-171-148.austin.ibm.com
Service name          = 50000
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select * from author"
SQL30108N  A connection failed in an automatic client reroute environment. The
transaction was rolled back. Host name or IP address:
"ausipas066.austin.ibm.com". Service name or port number: "50000". Reason
code: "1". Connection failure code: "2". Underlying error: "32".
SQLSTATE=08506
```

Upon encountering SQL30108N, the DB2 client reestablishes a connection with the new primary ITSO database. Example 5-47 demonstrates this by obtaining the connection state of the DB2 client again. The information from the client now confirms that it is connected to the new primary ITSO database. As a result, submitting the same query again succeeds. This proves that you have successfully performed a manual takeover of the ITSO database.

Example 5-47 DB2 client has re-established a connection with the new primary ITSO database

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state
```

Database Connection State

```
Connection state      = Connectable and Connected
Connection mode       = SHARE
Local database alias  = ITSO
Database name         = ITSO
Hostname              = ausipas066.austin.ibm.com
Service name          = 50000
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "SELECT * FROM AUTHOR"
```

```
AUTHOR_NUMBER  AUTHOR_NAME
-----
```

```
1 Margaret Ticknor
```

```
1 record(s) selected.
```

5.8.2 Unplanned outage - shutdown of primary database OS

This scenario simulates an *unplanned* outage of the primary database server. This can happen, for example, when a Compute Node in the Cloud Group where the Virtual System Pattern is deployed suffers a (fatal) hardware failure. This scenario demonstrates that a DB2 client connected to the HADR database is not affected by this example.

Establish a connection and commit transaction from DB2 client

This scenario starts where 5.8.1, “Planned outage: DB2 takeover” on page 165 ended. You should still have a connection established from the primary ITSO database on ausipas066.austin.ibm.com. Example 5-48 shows how you **commit** another transaction using that connection. The **commit** to the primary ITSO database synchronizes with the standby ITSO database before returning the DB2 client (auto) **commit** call.

Example 5-48 Committing a transaction over the existing DB2 client connection

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state

Database Connection State

Connection state      = Connectable and Connected
Connection mode      = SHARE
Local database alias = ITSO
Database name        = ITSO
Hostname             = ausipas066.austin.ibm.com
Service name         = 50000
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "INSERT INTO AUTHOR (AUTHOR_NUMBER,
AUTHOR_NAME) VALUES (2, 'Kyle Brown')"
```

DB20000I The SQL command completed successfully.

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "SELECT * FROM AUTHOR"
```

| AUTHOR_NUMBER | AUTHOR_NAME |
|---------------|------------------|
| 1 | Margaret Ticknor |
| 2 | Kyle Brown |

2 record(s) selected.

Shutdown primary DB2 server

Example 5-49 shows existing DB2 connections on the primary DB2 server.

Example 5-49 List existing DB2 connections on the primary DB2 server

```
[db2inst1@ausipas066 ~]$ db2 list applications
```

| Auth Id | Application | Appl. | Application Id |
|---------|-------------|--------|------------------------------|
| DB | # of | | |
| Name | Name | Handle | |
| | Agents | | |
| ITSUSER | db2bp | 252 | 9.3.171.4.35604.141022152438 |
| ITSO | 1 | | |

Example 5-50 shows a shutdown of the primary DB2 server operating system.

Example 5-50 Graceful shutdown of the primary DB2 server's Operating System

```
login as: root
root@9.3.169.66's password:
-bash-4.1# hostname -f
ausipas066.austin.ibm.com
-bash-4.1# shutdown -h now
```

```
-bash-4.1#  
Broadcast message from root@ausipas066  
(/dev/pts/2) at 15:48 ...
```

The system is going down for halt NOW!

With the primary DB2 server shutdown, you can confirm that the standby server has now become the new primary (see Example 5-51).

Example 5-51 Confirming the standby server is now the primary server

```
[db2inst1@ipas-lpar-9-3-171-148 ~]$ db2pd -hadr -database its0 | grep -E  
"HADR_ROLE|HADR_SYNCMODE|HADR_CONNECT_STATUS "  
          HADR_ROLE = PRIMARY  
          HADR_SYNCMODE = SYNC  
          HADR_CONNECT_STATUS = DISCONNECTED
```

Run SQL query from the DB2 client

Next, the scenario runs the same SQL query again using the DB2 connection to the database that was already established before the shutdown of the primary. As shown in Example 5-52, the DB2 client encounters condition SQL30108N and rolls back the transaction.

Example 5-52 DB2 client encounters a transaction rollback when performing a SQL query

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state
```

Database Connection State

```
Connection state      = Connectable and Connected  
Connection mode       = SHARE  
Local database alias  = ITS0  
Database name        = ITS0  
Hostname              = ausipas066.austin.ibm.com  
Service name         = 50000
```

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select * from author"  
SQL30108N A connection failed in an automatic client reroute environment. The  
transaction was rolled back. Host name or IP address:  
"ipas-lpar-9-3-171-148.austin.ibm.com". Service name or port number: "50000".  
Reason code: "1". Connection failure code: "1". Underlying error: "110".  
SQLSTATE=08506
```

After encountering the SQL30108N condition, the DB2 client should automatically re-establish a DB2 connection to the new primary DB2 server. This is demonstrated by performing another SQL query, which completes without any issues. Note that this scenario did not establish a new connection, for example, you do not have to authenticate again (see Example 5-53).

Example 5-53 Upon submitting the same SQL query again the DB2 transaction completes as expected

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 get connection state
```

Database Connection State

```
Connection state      = Connectable and Connected  
Connection mode       = SHARE  
Local database alias  = ITS0
```

Database name = ITS0
Hostname = ipas-lpar-9-3-171-148.austin.ibm.com
Service name = 50000

```
[db2inst1@ipas-lpar-9-3-171-4 ~]$ db2 "select * from author"
```

```
AUTHOR_NUMBER  AUTHOR_NAME  
-----  
1 Margaret Ticknor  
2 Kyle Brown
```

```
2 record(s) selected.
```



High availability and disaster recovery scenarios for WebSphere Application Server

This chapter describes how to implement scenarios that use GPFS as a persistent store for transaction data. Chapter 3, “High availability and disaster recovery scenarios” on page 29 describes these scenarios at a high level.

The WebSphere Application Server transaction service writes information to a transaction log for every global transaction that involves two or more resources, or that is distributed across multiple servers. The transaction service maintains transaction logs to ensure the integrity of transactions. It is essential to store transaction logs in a sophisticated file system such as the IBM General Parallel File System (GPFS).

This chapter covers the following scenarios:

- ▶ Scenario WAS_1: WebSphere cell in the same rack, transactions in GPFS
- ▶ Scenario WAS_2: Single WebSphere Cell Across two racks in PDC
- ▶ Scenario WAS_3: Active/Passive, identical setups in PDC and SDC, with transactions stored in GPFS

6.1 Scenario WAS_1: WebSphere cell in the same rack, transactions in GPFS

This scenario involves a primary GPFS server and a WebSphere Application Server cluster. Both are running in a single rack. WebSphere Application Server stores its transaction logs in the file system that is managed by the GPFS server. The following are the steps that are required to implement this scenario, which is shown in Figure 6-1:

1. Configure and deploy GPFS as a primary server.
2. Deploy a Shared Service for GPFS in a cloud group.
3. Configure a single cell WebSphere Application Server.
4. Add GPFS Client policy to WebSphere Application Server cell.
5. Deploy WebSphere Application Server pattern.
6. Test the HA scenario.

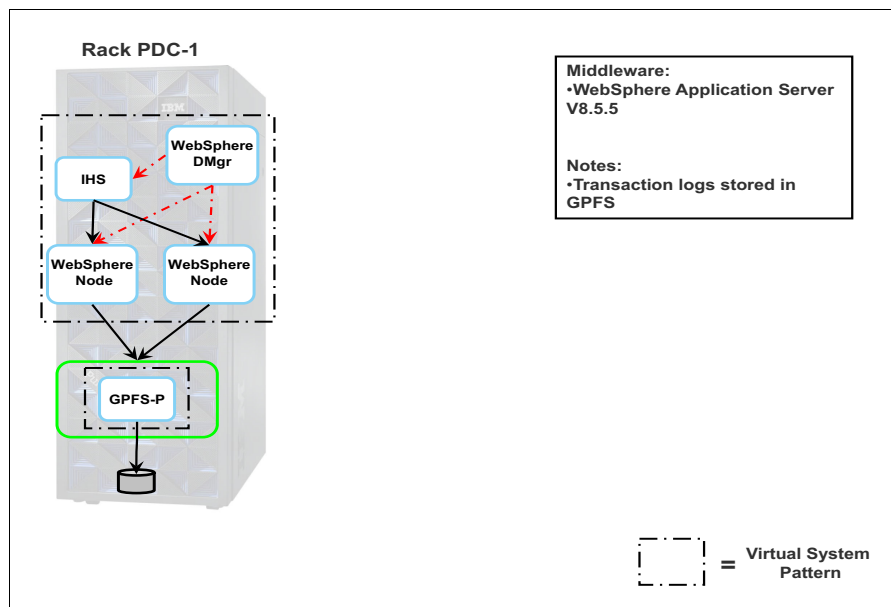


Figure 6-1 Scenario WAS_1

6.1.1 Configure Primary GPFS Server

This scenario involves only a single primary GPFS server. If a more sophisticated GPFS cluster is needed, you can add a mirror server and a tiebreaker server in the same rack or in separate racks. To start a single primary GPFS server, complete the following steps:

1. Allocate block storage volumes
Follow the instructions in 4.1.1, “Block storage configuration” on page 48 to create a storage volume for the GPFS server.
2. Create a GPFS virtual application pattern of pattern type, GPFS Pattern Type 1.2
Follow the instructions in 4.3.1, “Active/Active GPFS deployment: Steps” on page 53 to configure a Primary GPFS server. That section covers more than what is required for this scenario. There is no need to configure a mirror or a tiebreaker.
3. Deploy the GPFS pattern that was created in step 2 with the storage volume created in step 1 allocated to it. Follow the instructions in 4.3.1, “Active/Active GPFS deployment: Steps” on page 53 for the deployment procedure.

4. Create a Shared Service for GPFS

A shared service for GPFS instance provides linkage between GPFS server and its clients in the same cloud group. The shared service for GPFS holds the private key to the GPFS server. In version 2.0, PureApplication System allows only one such shared service for GPFS per cloud group. See 4.5, “Deploy GPFS Shared Service” on page 63 for detailed procedures.

6.1.2 Build the WebSphere Application Server cluster pattern

For information about this process, see “Build the WebSphere Application Server cluster pattern” in Appendix B, “Common WebSphere Application Server configuration tasks” on page 265”.

The pattern name for this scenario can be set to, for example, RB_HADR_WAS_1a.

6.1.3 Add GPFS Client Policy

Open the WebSphere Application Server pattern in the pattern builder. Use the following steps to add a GPFS Client Policy to custom nodes:

1. Add a GPFS Client Policy to each node of the WebSphere Application Server pattern that needs to use the block storage. For this scenario, work on the custom nodes and optionally the deployment manager.
 - a. Open the pattern builder for the WebSphere Application Server pattern.
 - b. Locate the custom node.
 - c. On the Custom Node component, click **Add a Component Policy**. It is the first icon with a + on the left.

After the policy is added, the GPFS Client Policy box is added at the bottom of the CustomNode. See Figure 6-2.

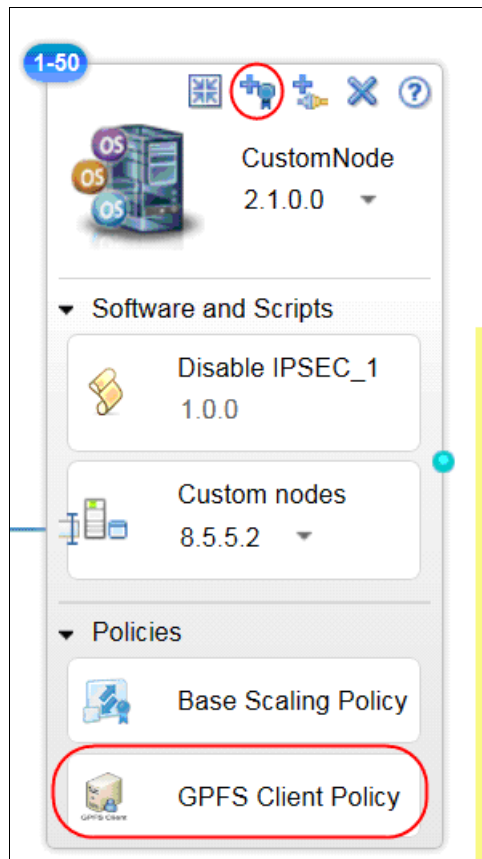


Figure 6-2 GPFS Client Policy

2. Configure the GPFS Client Policy.:
 - a. Every GPFS Client is associated with a GPFS Server and a block storage that is mounted to that GPFS server. When configuring a GPFS client, it is important to specify the correct mount point of the block storage that the GPFS client is associated with. The mount point is called the File System in a GPFS client. The GPFS Server *Get Cluster Status* operation provides a list of defined file systems and associated file sets. Another way to find a list of mount points of a GPFS server is by listing the GPFS Server instance from PureApplication systems administration console. Select **Workload Console** → **Instances** → **Virtual Applications** → **the GPFS server instance** → **Virtual machine perspective** → **GPFSserver-Main**. Scroll down to the Block storage section. Under the Mount Point column, RBHADRWAS_4 in the file path (/gpfs/RBHADRWAS_4) is the mount point. It is the file system required to define a GPFS Client, see Figure 6-3 on page 177.

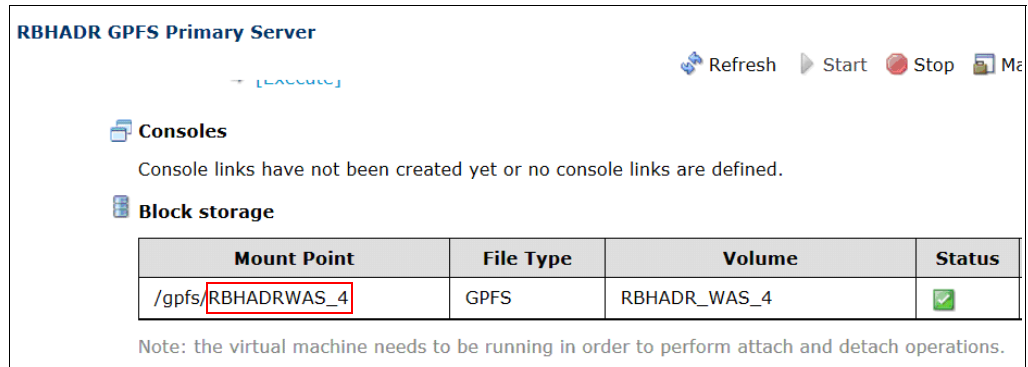


Figure 6-3 GPFS File System name

- b. Specify the File System information using the mount point found in the previous step. In this example, it is RBHADRWAS_4.
- c. Specify a directory and file set, which are intended for the customNode. The GPFS client creates a symbolic link with the name specified in the *Directory to link on local system* field.
- d. Specify a slash to put the symbolic link to the root directory. See Figure 6-4. If a slash is not provided as part of the link, an error is issued.

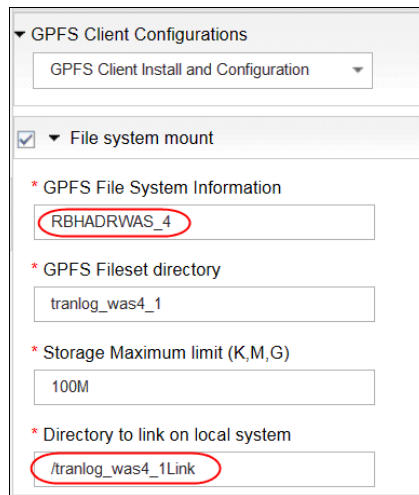


Figure 6-4 GPFS Client Policy configuration

6.1.4 Deploy WebSphere Application Server cluster pattern

After the WebSphere Application Server pattern is configured with the GPFS client policy, deploy and test it with an application. A test application was created for this publication. Follow the instructions in Appendix A, “Sample Application” on page 253 to install the test application after WebSphere Application Server is running.

The following is how to configure WebSphere Application Server so that each custom node in the cell writes its transaction log to the shared block storage managed by GPFS. Follow the instructions in “Deploy WebSphere Application Server cluster pattern: Single rack” on page 269 in Appendix B, “Common WebSphere Application Server configuration tasks” on page 265.

The pattern name GPFS used for this scenario is RB HADR WAS 1a.

6.1.5 Create a WebSphere Application Server cluster

To create the WebSphere Application Server cluster, follow the instructions in “Create WebSphere Application Server cluster” in Appendix B, “Common WebSphere Application Server configuration tasks” on page 265.

6.1.6 Configure transaction services

Use the following steps to configure the transaction services:

1. Start the Deployment manager's (DMgr) administrative console.
2. From the DMgr administrative console, select **Servers** → **Server Types** → **Application servers**.
3. Select server name **ClusterMember1**.
4. From the right pane, select **Container services** → **Transaction services**.
5. In the Transaction log directory, enter the full path including the leading slash of the symbolic link generated by GPFS client. In this example, `/tranlog_was4_1Link`.
6. Add a subdirectory, `member1`, under this folder. See Figure 6-5.

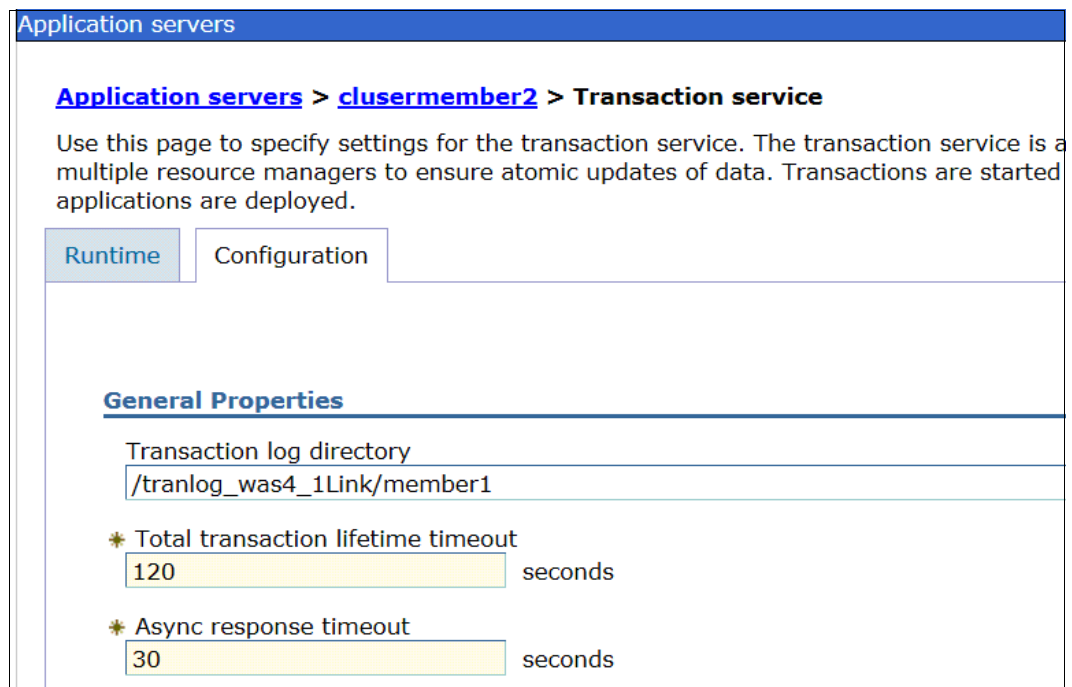


Figure 6-5 Configuring the transaction service

6.1.7 Test Scenario WAS_1 HA

Now that the GPFS server, shared service for GPFS, and WebSphere Application Server are running, run a few transactions and kill one of the cluster members, which in this example is ClusterMember1 or ClusterMember2.

See the instructions in “Validate BankTransaction application” on page 284.

6.2 Scenario WAS_2: Single WebSphere Cell Across two racks in PDC

This scenario illustrates how to use two racks in the primary data center (PDC) to achieve high availability using GPFS. The storage volume that stores important data such as WebSphere Application Server's transaction logs in the PDC-1 rack can be replicated synchronously to PDC-2 rack for use as a backup during a PDC-1 outage. With the multi-domain deployment capability, you can deploy a WebSphere cell with some of the cluster members on another rack in the vicinity. Create a primary GPFS server on the PDC-1 rack and its mirror server on the PDC-2 rack. Along with a Tiebreaker server, the three GPFS servers make up an Active/Active GPFS cluster. See the topology diagram in Figure 6-6.

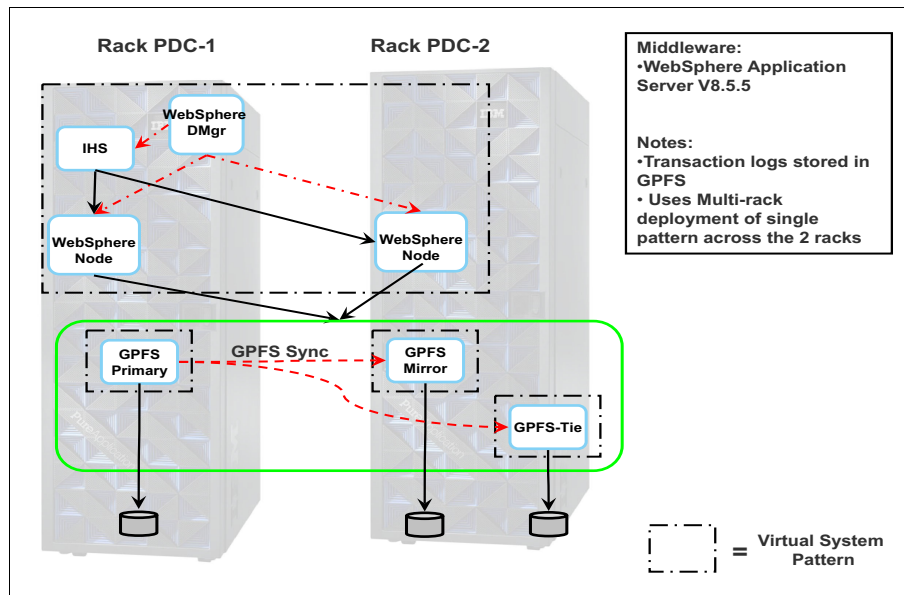


Figure 6-6 Single WebSphere Cell across two racks

The following is a list of the required steps:

1. Configure and deploy GPFS Mirror server at PDC-2.
2. Configure and deploy GPFS Tiebreaker server at PDC-2.
3. Configure and deploy GPFS Primary server at PDC-1.
4. Deploy GPFS Shared Service at PDC-1.
5. Configure WebSphere Application Server with GPFS client policy.
6. Deploy WebSphere pattern to multiple domains.
7. Configure WebSphere Application Server to write transaction log to GPFS storage volume.
8. Test the HA scenario.

6.2.1 Configure and Deploy GPFS Mirror Server at PDC-2

For more information about how to deploy a mirror GPFS server, see 4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53. The major steps required are listed below:

1. Create a storage volume for the mirror GPFS server as shown in Figure 6-7:
 - a. From the system console, select **Cloud** → **Storage Volumes** → **Create New**.
 - b. Create a 10 GB block storage volume named RBWAS2b_mirror. The storage size must match the target GPFS cluster’s primary GPFS server’s storage size. Select the correct cloud group.

Create a new storage volume

* Name:

Description:

Type:

Storage Volume Group:

* In Cloud Group:

Volume configuration: Existing settings Customize settings

* Size:

Figure 6-7 Create Storage Volume for Mirror Server

2. Deploy to the mirror GPFS server. If there is not a mirror GPFS server pattern, follow the instructions in 4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53 to create one.
 - a. From the workload console, select **Patterns** → **Virtual Applications** → **Virtual Applications**.
 - b. Deploy Mirror GPFS Server pattern:
 - i. Name: RBHADR GPFS Mirror Server
 - ii. File System Name for Selected Shared Volume: RHWAS2a (see Figure 6-8 on page 181)

Note: The File System Name should be the same for the Tie Server and Primary Server although each of them has their own storage volume.

Figure 6-8 File System for Mirror Server

3. Select a Storage Volume:
 - a. Click **Continue to Distribute**.
 - b. Move cursor to **Mirror Server Manager 1 VM**.
 - c. Click **Edit**.
 - d. Click the Storage Volume page.
 - e. Select the shared volume created for this server, **RHWAS2a_mirror**, as shown in Figure 6-9.
 - f. Click **Deploy**.

Figure 6-9 Create Storage Volume for Mirror Server

6.2.2 Configure and Deploy Tiebreaker Server at PDC-2

The procedure to deploy a Tiebreaker Server is similar to the steps in 6.2.1, “Configure and Deploy GPFS Mirror Server at PDC-2” on page 180.

1. Create storage volume for Tiebreaker Server, see Figure 6-10.

Note: The storage volume size of Tiebreaker can be smaller than the Mirror server. Select the correct cloud group.

Create a new storage volume

* Name:

Description:

Type:

Storage Volume Group:

* In Cloud Group:

Volume configuration: Existing settings Customize settings

* Size:

Figure 6-10 Create Storage Volume for Tiebreaker

2. Deploy tiebreaker GPFS server. The File System name mapped to the storage volume should be the same as the Mirror server, see Figure 6-11.

GPFS Tie Breaker Server

Fill in the required values for component **GPFS Tie Breaker Server** for this pattern.

Key Management

GPFS Managers Key:

GPFS Cluster Key:

GPFS Clients Key:

GPFS Configurations

* File System name for selected shared volumes:

Figure 6-11 File System for Tiebreaker

3. Select the storage volume. See Figure 6-12 for specific values.

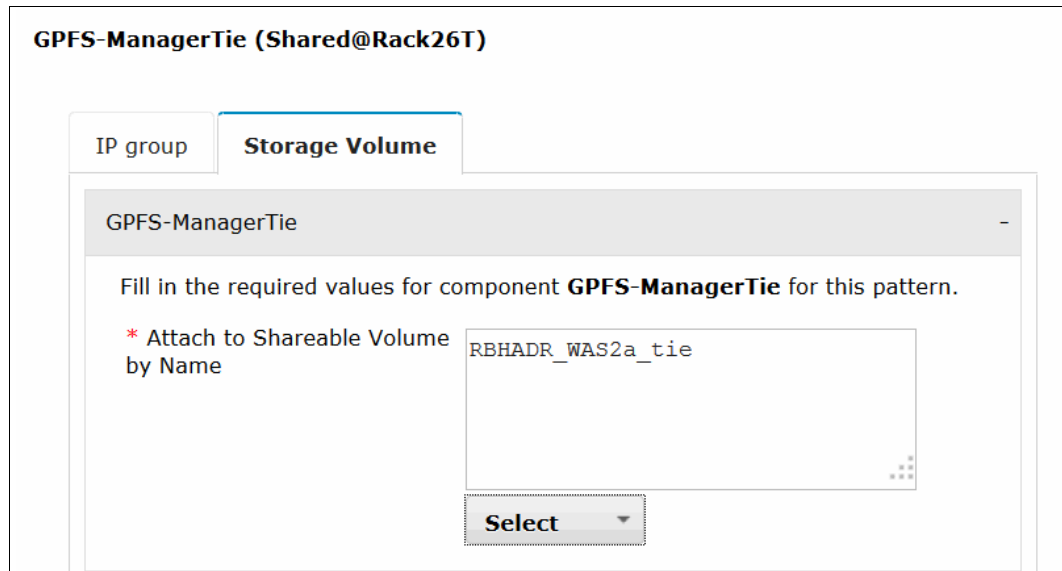


Figure 6-12 Select Storage Volume for Tiebreaker

6.2.3 Configure and deploy GPFS Primary server on PDC-1

In this scenario, the Mirror server and Tiebreaker server are deployed before deploying the Primary server. You can deploy the Primary pattern with the Mirror Server and Tiebreaker's manager VM addresses within it. See Figure 6-13 on page 184.

Follow the instructions in 4.3.1, "Active/Active GPFS deployment: Steps" on page 53 to configure the GPFS Primary server on PDC. The main steps are highlighted below:

1. Create a storage volume for the Primary GPFS Server:

The storage volume size of Primary must be the same as the Mirror server. Select the correct cloud group.

2. Deploy Primary GPFS server:

The File System name mapped to the storage volume should be the same as that for the Mirror and Tiebreaker.

▼ GPFS Configurations

Primary Configuration ▼

* Number of GPFS Node(s) 1 node ▼

* Cluster Name Primary_Cluster

* File System name for selected shared volumes RBWAS2a

* Snapshot Interval (minutes) 60

▼ Active Configuration (Mirror and Tie)

Mirror Manager IP 172.20.64.7

Tiebreaker Manager IP 172.20.64.15

Figure 6-13 Deploy Primary GPFS Server

3. Include Mirror and Tiebreaker manager's IP address as the cluster members:
 - a. Expand **Active Configuration (Mirror and Tie)**.
 - b. Enter the IP address of the Mirror server and Tiebreaker server.
4. Select the storage volume as shown in Figure 6-14.

GPFS-Manager (RBHADR_Internal_CG@Rack33)

IP group **Storage Volume**

GPFS-Manager -

Fill in the required values for component **GPFS-Manager** for this pattern.

* Attach to Shareable Volume by Name RBHADR_WAS_2a_primary

Select ▼

Figure 6-14 Select Storage Volume

6.2.4 Deploy GPFS Shared Service at PDC-1

Deploy the GPFS Shared Service after the Primary GPFS server is running. To complete the deployment of GPFS Shared Service, use these steps:

1. Obtain the SSH Client key from the Primary GPFS server instance (Figure 6-15):
 - a. From the workload console, select **Instances** → **Virtual Applications** → **RBHADR GPFS Primary Server**.
 - b. On the right pane, click **Manage**.
 - c. Stay on the right pane with **GPFS-Manager** highlighted. Scroll down to **Manage Keys**.
 - d. From the **Key Type** list, select **Client**.
 - e. Click **Submit**. Wait for the operation complete.
 - f. Check the result by clicking **Report**. This action opens a web page that displays the content of the key.
 - g. Select all text including the IP address, key text, and comment.
 - h. Copy all text to the clipboard. This text is used when to start the GPFS Shared Service in the next step.

| Name | Status | Created Time | Result | Return Value |
|--------------|--------|--------------------------|--|--|
| Retrieve Key | Done | Oct 26, 2014, 7:31:27 AM | GPFS-Manager.11414205368216.C Success | GPFS-Manager.11414205368216.GPFS_Manage Client Key successfully collected. : client private key |

Figure 6-15 Retrieve Key Operations and resulting key

2. Deploy the GPFS Shared Service (Figure 6-16):
 - a. From the workload console, select **Cloud** → **Shared Services**.
 - b. Expand **IBM Shared Service for GPFS**.
 - c. Select **IBM Shared Service for GPFS (External) 1.2.0.0**.
 - d. Click **Deploy**.
 - e. Paste the SSH client key obtained in the previous step.
 - f. Select the correct Environment Profile. In this scenario, choose the profile that can serve the multi-domain component. In this case it is the environment profile: **Multitrack Content**
 - g. Click **Continue to distribute**.

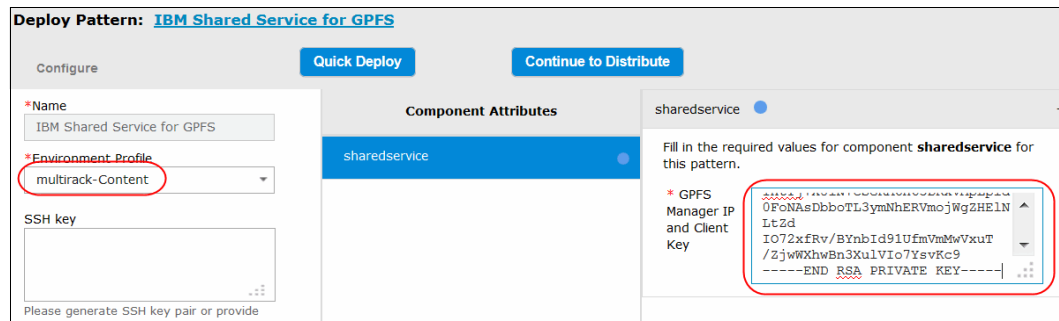


Figure 6-16 Shared Service for GPFS

6.2.5 Configure WebSphere Application Server with GPFS client policy

First, create a WebSphere Application Server cluster pattern. Next, add a GPFS client policy to a custom node. Complete the following steps:

1. To create the WebSphere Application Server cluster pattern, see the instructions in “Build the WebSphere Application Server cluster pattern” on page 266.
2. To add the GPFS client policy, follow the instructions in 6.1.3, “Add GPFS Client Policy” on page 175.

6.2.6 Deploy WebSphere pattern to multiple domains

Next, deploy the WebSphere Application Server cell to two racks. One custom node is on the PDC-1 and the other node is on the PDC-2. Both custom nodes write their transaction logs to the same storage volume. The storage volume is on the PDC-1 and is mirrored and copied synchronously to the storage volume that is on the PDC-2. In the event of a storage volume outage on PDC-1, both custom nodes are automatically rerouted to the mirror GPFS server to serve the data retrieved from the mirrored data store managed by the mirror GPFS server.

To deploy your pattern to multiple domains, your ID must be authorized to use the environment profile for multiple domain. Your login ID should also be authenticated by an LDAP server. Ensure that your environment has been configured to enable multiple domain deployments. Listed here are the high-level steps to deploy a pattern with custom nodes scattered across multiple domains. Complete deployment instructions are in “Deploy WebSphere Application Server cluster pattern: Multiple rack” on page 270.

To deploy the WebSphere pattern to multiple domains, complete these steps:

1. Log in to PureApplication System using your user ID and password for multiple domain user group.
2. Deploy the pattern the same way as when you deploy locally.
3. Choose the environment profile that is set for multiple domain deployment. This configuration separates management and data networks as shown in Figure 6-17.

The screenshot shows a 'Configure' window with the following fields:

- *Name:** RBHADR_WAS_2a_GPFS
- *Environment Profile:** multirack-Content
- *Priority:** High

Figure 6-17 Environment Profile for WebSphere Application Server GPFS Client

4. At deployment time, select the domain to place each VM. Make the decision based on the VM's location and deploy it. See Figure 6-18 for an example. One custom node is placed on Rack 26T while all other nodes are deployed to Rack 33, see Figure 6-18.

The screenshot shows the 'Deploy Pattern' interface for 'RBHADR WAS 2a GPFS'. It includes a 'Deploy' button and a table showing the distribution of components across racks.

| Components | Rack26T | Rack33 |
|--------------------------|---------|-----------|
| | Public | Public-33 |
| IHSNode (1 VM) | | 1 VM |
| DmgrNode (1 VM) | | 1 VM |
| CustomNode (2 VMs, 1-50) | 1+ VM | 1+ VM |

Figure 6-18 Put the VM in the rack of your choice

6.2.7 Configure WebSphere Application Server to write transaction log to GPFS storage volume

For instructions, see 6.1.6, "Configure transaction services" on page 178.

6.2.8 Test the Multi-domain WebSphere Split Cell HA using GPFS scenario

The storage volume that is managed by GPFS can be replicated from one rack to another rack in the vicinity.

WebSphere Application Server stores its transaction logs on the volume that is managed by the primary GPFS server on PDC-1. The transaction logs are copied over to the file system managed by the mirrored server on PDC-2. The Tiebreaker server monitors the health of both the primary and mirror servers. It decides which server should render the content of the file systems to GPFS clients on either rack. It is transparent to those GPFS clients who are using the data in the event of losing one of the GPFS servers or one of the storage volumes.

By splitting WebSphere Application cell across two racks, the application server can continually take workload in case one of the cluster members goes down. After the GPFS server cluster is up and WebSphere Application Server is running, use these steps to test this scenario:

1. On PDC-1, stop the Primary GPFS server:
 - a. From Workload console, select **Instances** → **Virtual Applications**.
 - b. Select **RBHADR GPFS Primary Server**.
 - c. Click **Stop** on the right pane. Wait for the stop operation to complete.
2. Test the application. The mirror server should take over the file serving work from the Primary server that is down. Nothing needs to be done on PDC-2 to recover the outage of PDC-1. The outage of PDC-1 should be transparent to WebSphere Application Server. For more information, see “Validate BankTransaction application” on page 284.

6.3 Scenario WAS_3: Active/Passive, identical setups in PDC and SDC, with transactions stored in GPFS

This scenario, shown in Figure 6-19, involves a pair of WebSphere Application Server cells, GPFS servers, and block storage volumes. In the PDC, the transaction log files of WebSphere Application Server are stored in the block storage managed by GPFS. The block storage is replicated either synchronously or asynchronously to another block storage that is at the SDC. The SDC is configured to take both a planned and an unplanned outage of the PDC.

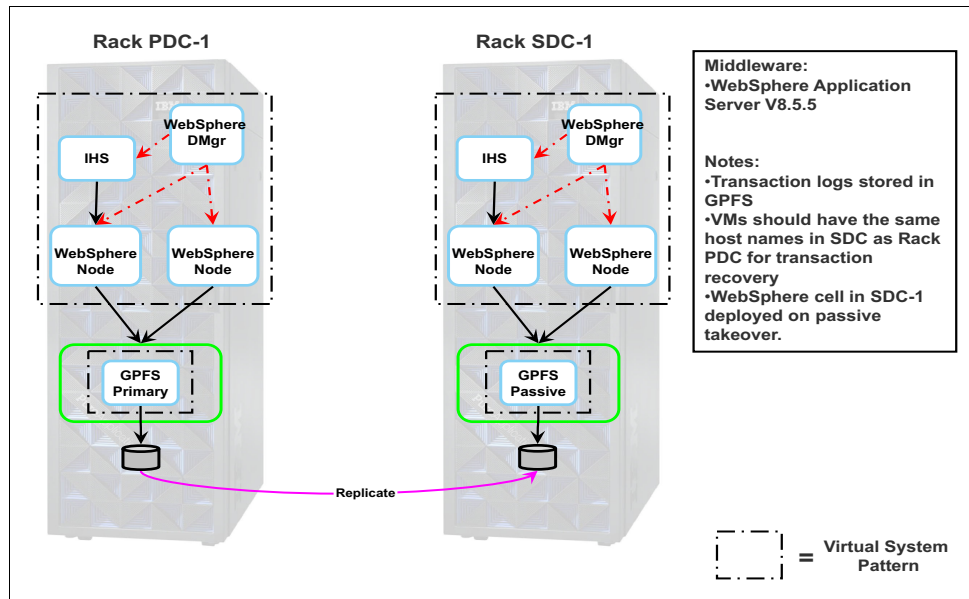


Figure 6-19 WebSphere active/passive cell setup

WebSphere transaction log recovery requires the application servers at the SDC have the same host names as those in the PDC. To avoid name resolution conflicts, special network configuration is needed. There are different ways to achieve this. For instance, manipulate the `/etc/hosts` to resolve local host names. Another way to achieve this is by setting up a special DNS on each site, PDC and SDC, to reserve and resolve the host name for all the WebSphere Application Server VMs. This configuration is described in detail in 4.9, “DNS setup for primary and secondary (cross) rack scenarios” on page 81.

The SDC is configured to recover both planned and unplanned outages of the PDC. Although the recovery procedure described in this scenario is for a planned outage, the same procedure that takes place at the SDC also works for an unplanned outage. Table 6-1 outlines the configuration steps and execution sequence starting from initiating all the components at the PDC and SDC to the recovery from SDC.

Table 6-1 Configuration steps

| Task | Configuration | Data Center |
|---------|--------------------------------|-------------|
| Network | Configure IP group used for DR | PDC |
| | Configure IP group used for DR | SDC |

| Task | Configuration | Data Center |
|-------------------------------------|--|--------------------|
| Cloud Group | Configure cloud group and environment profile for DR | PDC |
| | Configure cloud group and environment profile for DR | SDC |
| Storage Volume | Create primary storage volume | PDC |
| | Create standby storage volume | SDC |
| | Create primary Block Storage Replication Profile | PDC |
| | Create standby Block Storage Replication Profile | SDC |
| | Create storage volume replication pair | PDC |
| | Accept storage volume replication pair | SDC |
| GPFS | Deploy Passive GPFS Server | SDC |
| | Deploy Primary GPFS Server | PDC |
| | Add Passive GPFS Server as member of the Primary GPFS cluster | PDC |
| | Obtain Primary GPFS Server's client key | PDC |
| | Deploy GPFS Shared Service with the Primary GPFS Server's client key | PDC |
| WebSphere Application Server | Deploy WebSphere cell with GPFS client policy | PDC |
| | Configure WebSphere to store transaction logs to block storage | PDC |
| | Serving WebSphere applications | PDC |
| | Export WebSphere pattern | PDC |
| | Import WebSphere pattern | SDC |
| Planned Outage | Prepare Primary GPFS Server for failover | PDC |
| | Shutdown WebSphere | PDC |
| | | |

| Task | Configuration | Data Center |
|----------|--|-------------|
| Recovery | Storage Volume Replication Failover | SDC |
| | GPFS Server Takeover | SDC |
| | Obtain current GPFS Primary Server's client key | SDC |
| | Deploy Shared Service with the GPFS Server's client key | SDC |
| | Deploy WebSphere cell with GPFS client policy | SDC |
| | Configure WebSphere to store transaction logs to block storage | SDC |
| | Serving WebSphere Application Server applications | SDC |

6.3.1 Network configuration

For detailed steps, see 4.10, “Network configuration for WebSphere Application Server and DB2 scenarios” on page 85.

6.3.2 Cloud Group and Environment Profile

The cloud group to which the WebSphere Application Server VMs belong to must include a reserved IP group. All IP addresses in that IP group are reserved for WebSphere Application Server's VMs. The environment profile that WebSphere Application Server deployment uses must include that reserved IP group. These prerequisites must be met on both PDC and SDC.

PDC

In this example, the IPAD HADR IP group that is reserved for WebSphere Application Server VMs is included in the cloud group, RB DR Only. See Figure 6-20.

The screenshot shows the configuration for the 'RB DR Only' cloud group. Under the 'Deploy to cloud groups:' section, there are two tables. The first table lists cloud groups: RBHADR_Internal_CG (Private) with alias RBHADR_Internal_CG and a [remove] button. The second table lists IP groups: IPAS HADR (Data) with alias IPAS HADR and subnet address 172.20.88.0, and RBHADR_Internal_IPGroup (Data) with alias RBHADR_Internal_IPGroup and subnet address 172.20.88.0. The IPAS HADR row is highlighted with a red border, indicating it is selected for deployment.

Figure 6-20 Include IP Group in WebSphere environment profile

SDC

The cloud group requirement on SDC is the same. The cloud group must include the reserved IP group. The environment profile not only has to include the reserved IP group for the WebSphere Application Server deployment, but must also have the **Assign by deployer** option in the **IP addresses provided by** field, selected as shown in Figure 6-21.

The screenshot shows a configuration window titled "RB DR Only" with buttons for "Refresh", "Clone", and "Del". The "IP addresses provided by:" field is set to "Pattern Deployer" and is highlighted with a red box. Below it, the "Deployment priority:" is set to "Platinum - High(16) Medium(8) Low(4)", "Time zone:" is "Default time zone configured for the virtual image", and "Language:" is "Default language configured for the virtual image".

The "Deploy to cloud groups:" section contains a table with the following data:

| Name | Type | Alias | |
|-------------------------------------|-----------|--------|----------------|
| Shared | Private | Shared | [remove] |
| <input type="checkbox"/> | In use | Name | Alias |
| <input type="checkbox"/> | Share | Data | Share |
| <input checked="" type="checkbox"/> | IPAS HADR | Data | IPAS HADR |
| | | | Subnet address |
| | | | 172.20.64.0 |
| | | | 172.20.64.0 |

The row for "IPAS HADR" is highlighted with a red box.

Figure 6-21 Defining the IP Group

6.3.3 Storage Volume

Chapter 4, "Infrastructure setup" on page 47 lists step-by-step instructions to create storage volumes and block storage replication profiles on both PDC and SDC. Allocate a pair of similarly-defined block storage volumes on each side of the data center and verify that everything written to the primary storage is replicated to the storage volume on the passive site.

Creating storage volumes and creating block storage replication profiles are independent from each other. The block storage replication profile links a storage volume on SDC to a storage volume on the PDC. It defines which one is the source and which is the target of a replication action. It also provides a way to break the replication link or reverse the replication direction.

PDC

Create a storage volume on the PDC for the primary GPFS server first. Then, wait for the storage volume for the passive GPFS server to be created on the SDC. After that is done, create an entry in the block storage replication profile that is initiated from PDC by picking the storage volume for the passive GPFS server on SDC as the target. This action requires an acceptance from SDC, so wait for the acceptance to take place.

SDC

Create a stage volume on the SDC for the passive GPFS server. Notify the administrator on the PDC to initiate a block storage replication. Wait until the replication entry is created to accept the replication. Figure 6-22 identifies the button to accept or reject the replication request.

When a Passive GPFS Server is started with a storage volume allocated to it, the storage volume is not mounted until the passive node becomes the primary.

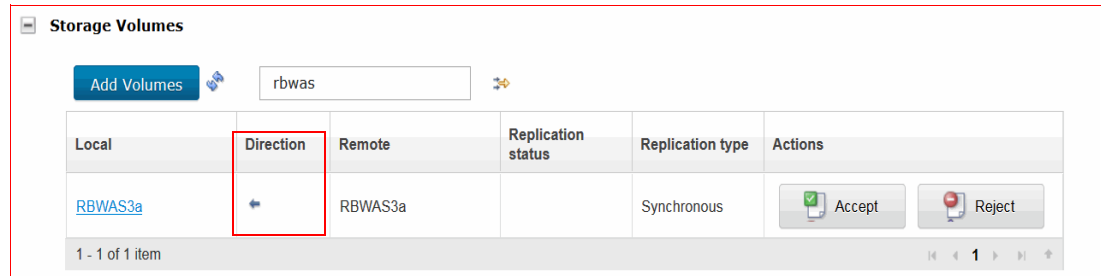


Figure 6-22 Accept Replication

Note: Be aware of the direction of the arrow in the Direction column. The arrow is pointing toward RBWAS3a, which means that the volume is receiving data from the Primary site.

6.3.4 GPFS

The sequence to start a GPFS Server starts with the Passive GPFS server. When the Passive GPFS server is started, the storage volume is allocated but not mounted. Therefore, the GPFS shared service cannot be started. Therefore, WebSphere Application Server GPFS clients cannot be deployed. WebSphere Application Server GPFS clients can only be deployed after the PDC is unavailable and the Primary takeover occurs.

PDC

The PDC must wait until the SDC starts its Passive GPFS server and obtains the Passive GPFS server's management VM IP address. That IP address is needed to add the Passive Server to the GPFS cluster. After the address is obtained, start the Primary GPFS Server and then start the GPFS Shared Service. After the GPFS Shared Service is started, deploy the GPFS clients.

SDC

The SDC site needs to deploy the Passive GPFS Server before the PDC. After the Passive server is running, supply the management VM IP address to the PDC. This is done so that the PDC can deploy its Primary GPFS Server or add the Passive Server as its Passive cluster member.

6.3.5 WebSphere Application Server

The WebSphere Application Server pattern can be deployed after the GPFS Shared Service is running. Each GPFS client relies on the GPFS Shared Service to obtain the access key to communicate with the GPFS Management node.

PDC

On the PDC side, WebSphere Application Server is deployed with the GPFS Client Policy included in the pattern. A list of all the host names of WebSphere Application Server cluster members is needed by the SDC administrator so that an identical WebSphere Application Server cluster can be brought up with matched host names. The IP assignment is automatically selected by the designated IP group at the deployment time. Table 6-2 shows an example of such a list:

Table 6-2 PDC WebSphere Cluster Host Name/IP list

| Node Type | Host Name | IP Address |
|-------------|---|--------------|
| CustomNode1 | ipas-hadr-034.purescale.raleigh.ibm.com | 172.20.95.34 |
| DmgrNode | ipas-hadr-035.purescale.raleigh.ibm.com | 172.20.95.35 |
| CustomNode2 | ipas-hadr-032.purescale.raleigh.ibm.com | 172.20.95.32 |
| IHSNode | ipas-hadr-038.purescale.raleigh.ibm.com | 172.20.95.38 |

After the connection to the block storage managed by GPFS is established, configure WebSphere Application Server to store its transaction log to the block storage. This is done so the transaction log can be replicated to the remote SDC's storage volume. Configure this for each custom node in the cell from the WebSphere Application Server's administrative console. Figure 6-23 shows where to configure WebSphere Application Server to write its transaction log to the mounted block storage.

Note: The file path that is specified for the transaction log directory field includes a symbolic link, `/tranlog_was4_1Link`, generated by GPFS. This symbolic link was specified in the GPFS Client Policy included in the WebSphere Application Server pattern.

Application servers

[Application servers](#) > [clusermember2](#) > **Transaction service**

Use this page to specify settings for the transaction service. The transaction service is a multiple resource managers to ensure atomic updates of data. Transactions are started applications are deployed.

Runtime Configuration

General Properties

Transaction log directory

* Total transaction lifetime timeout
 seconds

* Async response timeout
 seconds

Figure 6-23 Setting Transaction log to write to block storage

In this scenario, a test application, BankTransactionWeb, is installed. That application uses J2C alias to access databases through XA JDBC driver. The application requires J2C alias for security authentication. The J2C aliases must be identical on both PDC and SDC so that WebSphere Application Server can fully recover on the SDC. Create a list of the J2C aliases and provide it the SDC administrator. Follow these instructions to obtain the list:

1. Log in to WebSphere Application Server administration console, which is typically through the web link `http://dmgrIP:9060/ibm/console`. In this scenario, it is `http://172.20.95.35:9060/ibm/console` using ID: `virtuser` and the corresponding password.
2. From the WebSphere Administrative Console, select **Security** → **Global security** → **J2C authentication data**. Then, select each cluster member.

Figure 6-24 shows an example:

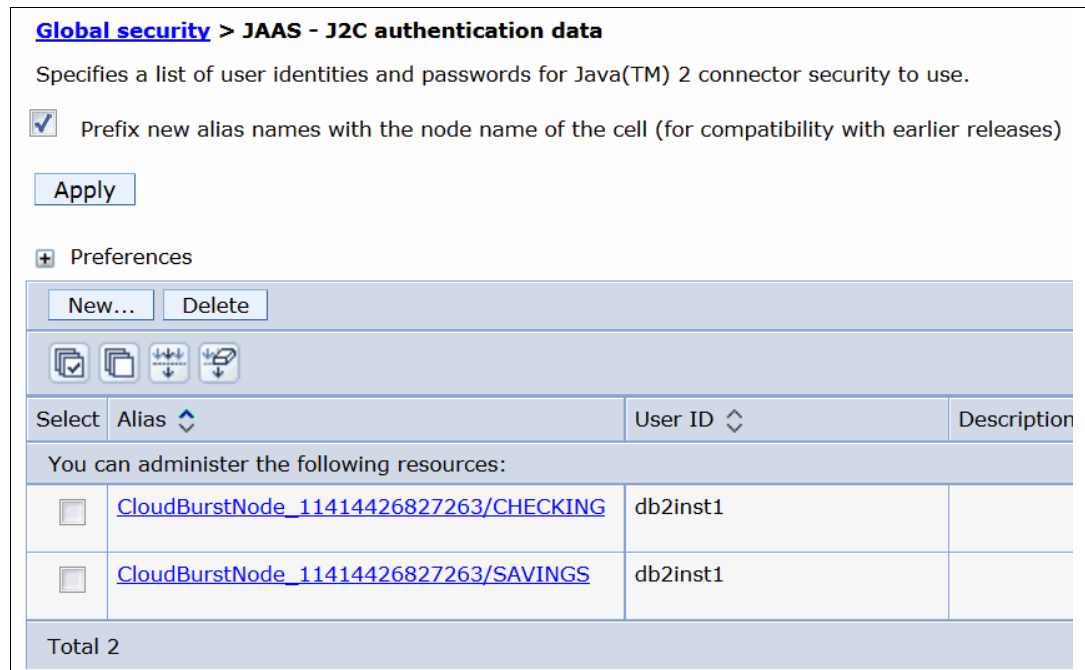


Figure 6-24 J2C Alias on PDC for database access

SDC

There is no action needed. Because the GPFS Server does not have the block storage volume mounted yet, there is no need to start its GPFS Shared Service. Therefore, you cannot start the WebSphere Application Server GPFS client. After the PDC is shut down, start the GPFS Shared Service, and then deploy WebSphere Application Server. In a real unplanned outage of the primary data center, the time to recovery is important. If WebSphere Application Server can be deployed and kept in standby mode before the outage of the primary occurs, the time to recovery can be shortened significantly.

To achieve this performance, deploy WebSphere Application Server ahead of time using the Shared Service for GPFS instance on PDC. After the WebSphere Application Server is running, stop its cluster members and put it in a standby mode. When the outage of PDC occurs, a new Shared Service for GPFS is started. At this point, the WebSphere Application Server's GPFS Client can be connected to the newly established Shared Service for GPFS instance on SDC.

Deploying WebSphere Application Server cluster pattern for this scenario requires extra steps to set up matching host names for all the WebSphere Application Server VMs that were deployed on PDC. See 4.9, “DNS setup for primary and secondary (cross) rack scenarios” on page 81 for how DNS is set up to keep the same host name for WebSphere Application Server cluster members. When planning DR, a set of host names is reserved on each data center as shown in Table 6-3.

Table 6-3 PDC WebSphere Cluster Host Name/IP list

| Host Name | IP Address |
|---|-------------------|
| ipas-hadr-001.purescale.raleigh.ibm.com | 172.20.95.1 |
| ipas-hadr-002.purescale.raleigh.ibm.com | 172.20.95.2 |
| ipas-hadr-003.purescale.raleigh.ibm.com | 172.20.95.3 |
| ... | ... |
| ipas-hadr-040.purescale.raleigh.ibm.com | 172.20.95.40 |

On SDC, the same range of IP are reserved for DR. Notice that the IP subnet is different whereas the host names on both sites are identical, as shown in Table 6-4.

Table 6-4 SDC WebSphere Cluster Host Name/IP list

| Host Name | IP Address |
|---|-------------------|
| ipas-hadr-001.purescale.raleigh.ibm.com | 172.20.71.1 |
| ipas-hadr-002.purescale.raleigh.ibm.com | 172.20.71.2 |
| ipas-hadr-003.purescale.raleigh.ibm.com | 172.20.71.3 |
| ... | ... |
| ipas-hadr-040.purescale.raleigh.ibm.com | 172.20.71.40 |

When deploying the WebSphere Application Server pattern on SDC, the IP address for each VM should not be automatically assigned by the IP group. Instead, the IP address should be assigned manually by the deployer because its host name needs to match its counterpart on PDC. Therefore, the IP group in the environment profile for this DR deployment is configured as “IP address provided by Pattern Deployer” as shown in Figure 6-25.

RB DR Only Refresh Clone De

IP addresses provided by: Pattern Deployer

Deployment priority: Platinum - High(16) Medium(8) Low(4)

Time zone: Default time zone configured for the virtual image

Language: Default language configured for the virtual image

Deploy to cloud groups:

| Name | Type | Alias | |
|---|---------|-----------|----------------|
| <input type="checkbox"/> Shared | Private | Shared | [remove] |
| <input type="checkbox"/> In use | Name | Alias | Subnet address |
| <input type="checkbox"/> Share | Data | Share | 172.20.64.0 |
| <input checked="" type="checkbox"/> IPAS HADR | Data | IPAS HADR | 172.20.64.0 |

Figure 6-25 IP address provided by Pattern Deployer

Deploy the WebSphere Application Server cluster pattern. Before assigning a VM and an IP address, you need to find the VM’s host name on PDC first. For instance, if the deploying VM is a deployment manager, DmgrNode, and if its host name is ipas-hadr-035.purescale.raleigh.ibm.com, as shown in Table 6-2 on page 194, then its IP address should be 172.20.71.35 as shown in Table 6-4 on page 196. The dialog box in Figure 6-26 must be filled as 172.20.71.35 for the DmgrNode. The Host Name field should be left blank as it is resolved by the DNS assigned to this IP group. The rest of the deployment process should be the same as the normal deployment process.

DmgrNode (Shared@Rack26T)

IP group

Virtual Machine 1

| | IP Group | Host name | IP Address (required) |
|------|-----------|-----------|-----------------------|
| eth1 | IPAS HADR | | 172.20.71.35 |

Figure 6-26 Assign IP Address

6.3.6 Planned outage at PDC

A planned outage gives the PDC a chance to gracefully shut down the GPFS server and other components. The overall goal is to put the primary GPFS Server in a position ready for the SDC GPFS server to take over its primary role. The Primary GPFS server must unmount the primary block storage and stop the replication from it to its counterpart at the SDC. To achieve this, complete the following steps:

1. Allow the primary GPFS Server to switch roles.
2. Fail over at the block storage replication.

From the workload console, complete the following steps:

1. Select **Workload** → **Instances** → **Virtual Applications**.
2. Locate the primary GPFS Server instance. On the right pane, click **Manage**.
3. On the right pane, under **AGENT** → **Operations**, highlight the **GPFS-Manager**.
4. There are several operations available as shown in Figure 6-27:
 - a. From the right pane, expand **Run Cluster Operation**.
 - b. Select **Prepare Primary for Takeover**.
 - c. Click **Submit**.
 - d. Wait for the result to return, which is shown in the lower part of the window.
 - e. To view the report, select **Result**. The report indicates that the role changed and the block storage was unmounted.

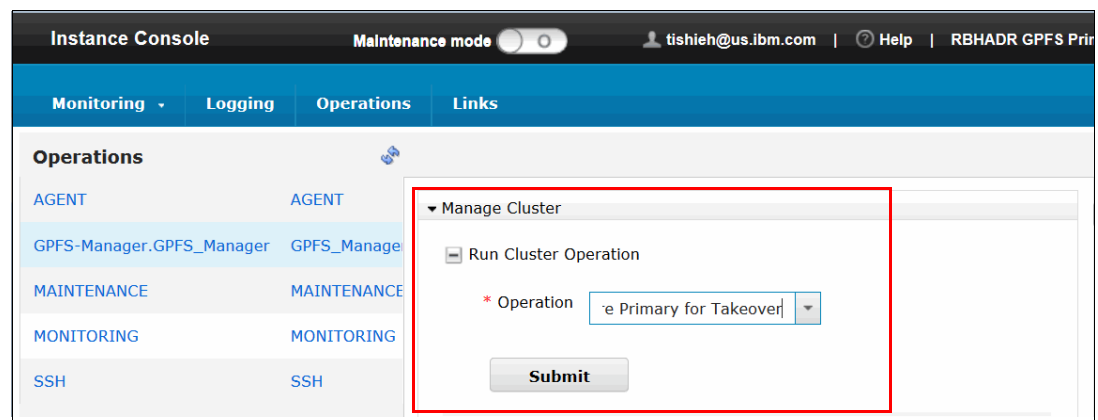


Figure 6-27 GPFS manager

After the block storage volume is released, go to the Block Storage Volume Replication to stop the replication.

1. From the system console, select **System** → **Block Storage Replication**.
2. Locate your block storage.

3. Select **Failover**. Notice that the arrow next to the block storage is pointing toward the passive volume. This indicates who the receiver of the data movement is. After you click **Failover**, the replication is stopped and the block storage is unlisted. An example is shown in Figure 6-28.

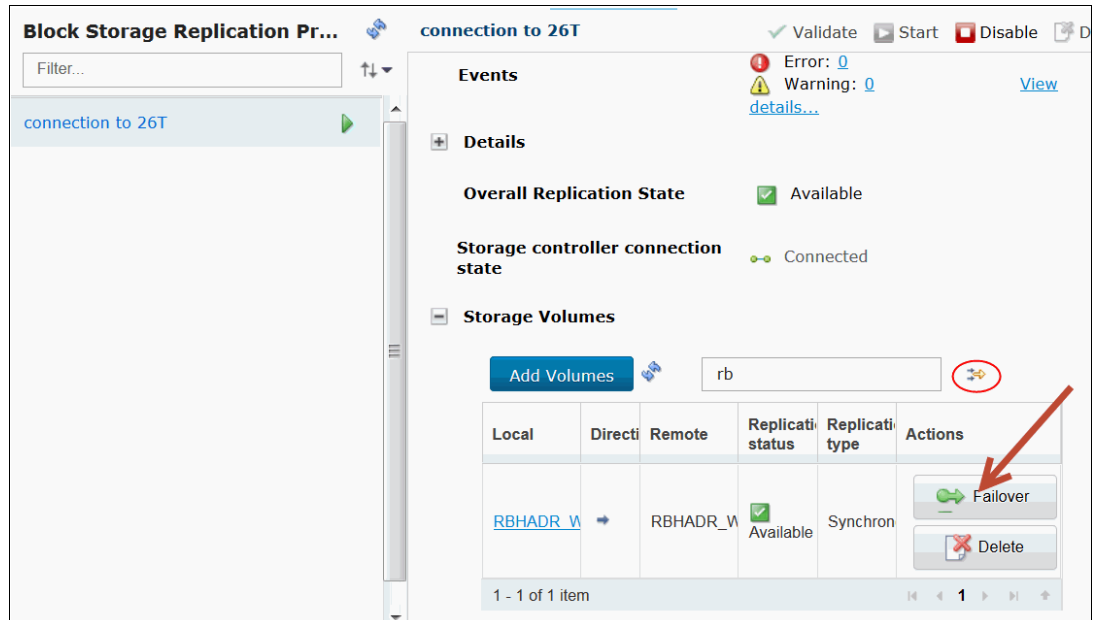


Figure 6-28 Storage Volume Replication Failover

After the primary GPFS Server switches the role and unmounts the block storage, WebSphere Application Server can no longer access the disk volume. You can either stop WebSphere Application Server or leave it running. If you leave it running, after the GPFS Server comes back to service, you can reconnect WebSphere Application Server (as GPFS Clients) to the GPFS Server. You can also restart the GPFS Shared Service so that it points to the newly recovered primary GPFS Server at the SDC. This scenario illustrates a complete shutdown of the PDC, so you cannot reconnect WebSphere Application Server at PDC to the GPFS Server at SDC.

6.3.7 Recovery

The recovery effort takes place at the SDC after PDC is shut down. Recovery requires two steps. First, switch the Passive GPFS Server's role by taking over the primary. Second, start a GPFS Shared Service instance and make it ready to accept GPFS clients.

SDC

From PureApplication System console, complete these steps:

1. Select **Workload Console** → **Instances** → **Virtual Applications**.
2. Locate your **GPFS Passive Server**.
3. Click your GPFS server to get its details.
4. From the right pane, select **Manage** → **Manager Agent**.
5. From the right pane, expand **Run Cluster Operation**.

6. Select **Passive Takeover** and click **Submit**, see Figure 6-29.

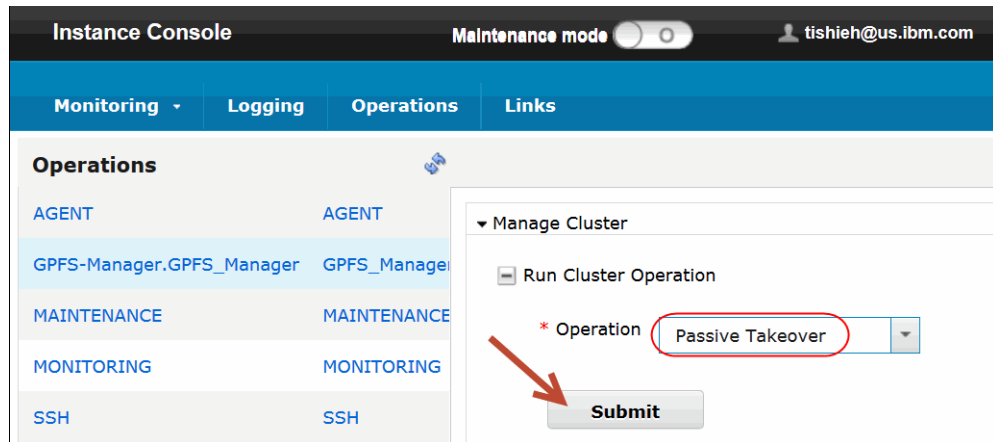


Figure 6-29 Switch the Passive GPFS server role

After the Passive Takeover task is complete, check the result from the lower pane. If no errors exist, get the client key from the current GPFS Server as shown in Figure 6-30.

1. Stay on the right pane with **GPFS-Manager** highlighted.
2. Scroll down to **Manage Keys**.
3. Select **Client** from the **Key Type** list.
4. Select **Submit**.

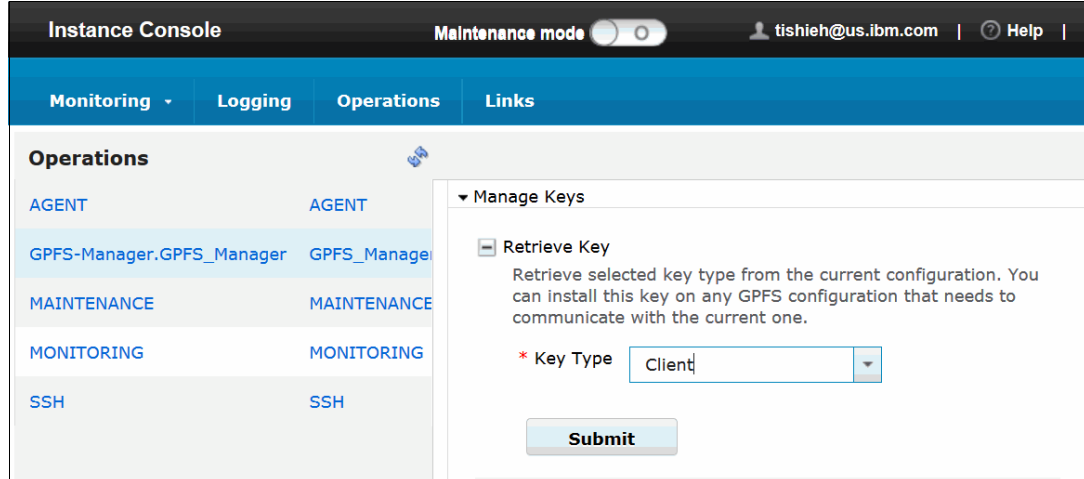


Figure 6-30 Client key

5. Wait for the operation to complete. Check the result by clicking **Report**, which opens a web page with the content of the key on display as shown in Figure 6-31.

The screenshot displays the WebSphere Operations console interface. At the top, there are navigation tabs: Monitoring, Logging, Operations, and Links. The 'Operations' tab is active, showing a list of operations on the left and a detailed view of the 'Retrieve Key' operation on the right. The detailed view includes a message box stating 'Operation "Retrieve Key" is launched' and a list of available operations: Retrieve Key, Install Key, Clear unused keys, Change Server GPFS Key, and Commit Server GPFS Key. Below this, the 'Operation Execution Results' section contains a table with the following data:

| Name | Status | Created Time | Result | Return Value |
|--------------|--------|--------------------------|--|--|
| Retrieve Key | Done | Oct 26, 2014, 7:31:27 AM | GPFS-Manager.11414205368216.G Success | GPFS-Manager.11414205368216.GPFS_Manage Client Key successfully collected. : client private key |

Figure 6-31 Retrieve client key

- Select all text including the IP address, key text, and comment and copy it to the clipboard as shown in Figure 6-32. This information is used to start GPFS Shared Service in the next step.



Figure 6-32 Private key

Start Shared services

The GPFS Shared Services must be started with the current Primary GPFS Server's client access key as shown in Figure 6-33.

From workload console, start the GPFS shared service:

1. Select **Cloud** → **Shared Services**.
2. Locate the GPFS Shared Service.
3. In the **Actions** column, select **Deploy**.

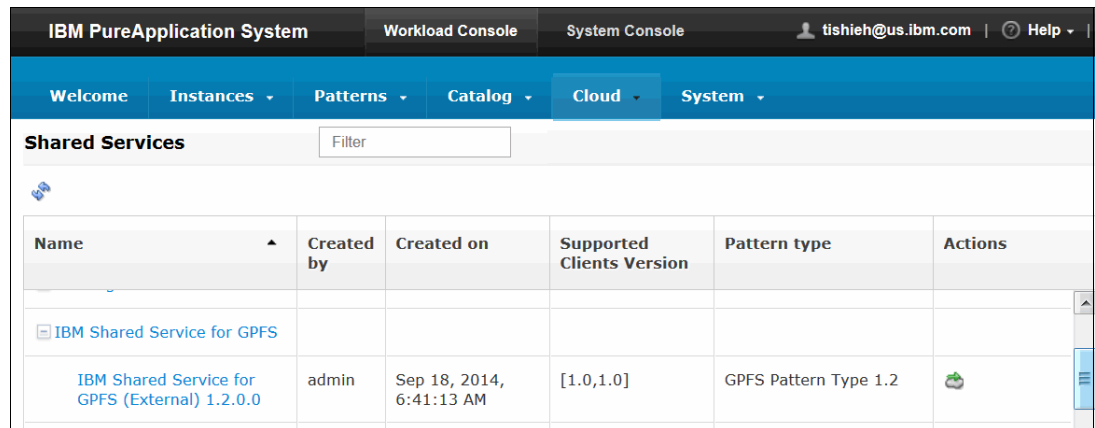


Figure 6-33 Shared services

4. After the Deploy window opens, paste the client key from the clipboard and deploy the Shared Service as shown in Figure 6-34.

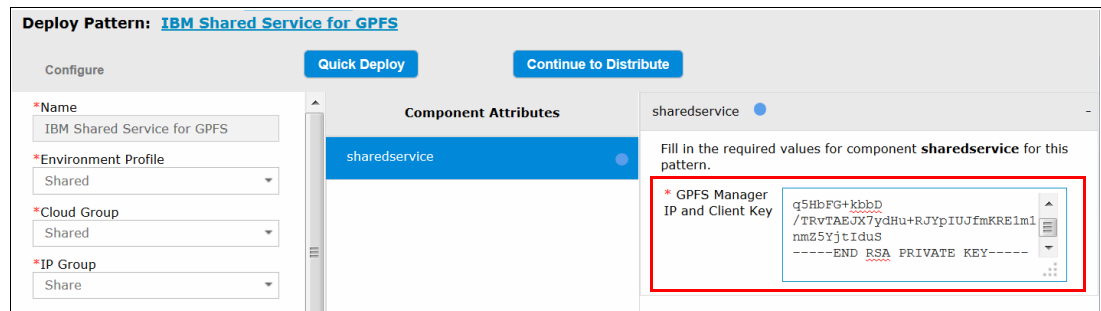


Figure 6-34 Paste client key

After the Shared Services is deployed, configure the WebSphere Application Server pattern with the same GPFS Client Policy specified earlier. Use the same File System and File System Link. After WebSphere Application Server instances are running, go to the WebSphere administrative console to configure its custom nodes to use the block storage for its transaction log as shown in Figure 6-35 on page 204.

1. From WebSphere administrative console, select **Servers** → **Server Types** → **WebSphere Application Servers**.
2. Enter the symbolic link that was generated by GPFS and add a subdirectory, such as **member1**, as the value of the Transaction log directory.

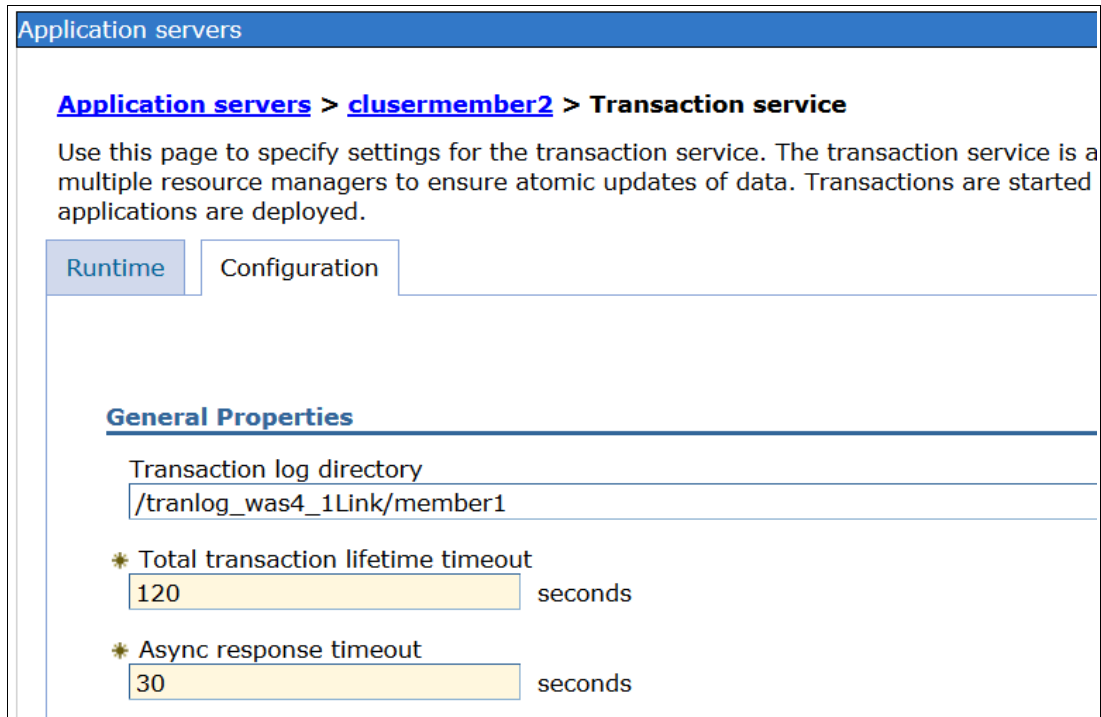


Figure 6-35 Use GPFS storage to store tranlog

To locate the two J2C alias at PDC, do the following as shown in Figure 6-24 on page 195. Issue WebSphere Application Server administrative commands to change the aliases at the SDC so they are the same as those at the PDC. Follow this procedure to do the change:

1. Open an SSH session to the Deployment Manager VM.
2. Go to /opt/IBM/WebSphere/ProfilesDefaultDmgr01/bin.
3. Issue the following command from the command line as shown in Example 6-1.

Example 6-1 Issuing the security = AdminConfig.getid command

```
security = AdminConfig.getid('/Cell:CloudBurstCell_11414620702085/Security:/')
print security
print AdminConfig.required('JAASAuthData')
alias = ['alias', 'CloudBurstNode_11414426827263/SAVINGS']
userid = ['userId', 'db2inst1']
password = ['password', 'passw0rd']
jassattrs = [alias,userid,password]
print AdminConfig.create('JAASAuthData', security, jassattrs)
AdminConfig.save()
```

4. After this code runs successfully, start the WebSphere Application Server Cluster and complete your recovery work.



High availability and disaster recovery scenarios for WebSphere Application Server and DB2

This chapter provides the details of the WebSphere Application Server scenarios in association with a DB2 database configured in a high availability and disaster recovery (HADR) configuration. This chapter specifically covers scenarios WDB_1, WDB_2, and WDB_3 as described in Chapter 3, “High availability and disaster recovery scenarios” on page 29. Although the steps for deploying WebSphere Application Server are described here, the steps for setting up DB2 are covered in Chapter 5, “High availability and disaster recovery scenarios for DB2” on page 89. The examples use WebSphere Application Server V8.5.5. software components that are included with the V2.0 pattern editor.

This chapter includes the following sections:

- ▶ Common assets used in scenarios
- ▶ Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database
- ▶ Scenario WDB_2: WebSphere Application Server cluster with DB2 HADR, split across two racks with WebSphere transactions stored in database
- ▶ Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in database

7.1 Common assets used in scenarios

Below are common assets that are used in the scenarios:

- ▶ Image parts

Core OS (IBM OS image for Red Hat Linux Systems 2.1.0.0 42): This is the default image that is used by all the pattern (Figure 7-1).



Figure 7-1 Core OS image

- ▶ Policies

Scaling Policy: Currently, the only thing the scaling policy is used for is to increase or reduce the number of nodes that are created on the component where it is enabled. In the examples, the custom node component is set to two, because those nodes make up the cluster members.

- ▶ Script packages and the parameters that are used in the pattern

Disable IPTables: This is a custom script that was written for this book. The “core OS” part used to build the patterns by default comes with all ports shut down. This script shuts down the firewall on each of the VMs so all ports are opened. This is not something that is recommended, but has been done here for the sake of simplicity. A better option is to have the script open only the ports that are necessary. This script is run at deployment time.

- ▶ Prerequisites

- Deployed DB2 environment

The patterns that make up the DB2 environment are described in Chapter 5, “High availability and disaster recovery scenarios for DB2” on page 89. Each of the scenarios uses an instance of DB2 HADR deployed by a different pattern. Both of the patterns, WDB_1 and WDB_2, use an environment deployed based on scenario DB2_1. The pattern WDB_3 uses an environment deployed based on concepts from both scenario DB2_2 and DB2_3.

This scenario requires that the following databases have been created as part of the DB2 HADR patterns deployed in Chapter 5, “High availability and disaster recovery scenarios for DB2” on page 89:

- TRANLOG: This is the database that will be used to recover transactions.
- CHECKING: Application database used by the application used in the scenario
- SAVINGS: Application database used by the application used in the scenario

- Application: The application used in the scenario, the details of which are provided in Appendix A, “Sample Application” on page 253.

7.2 Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database

This scenario covers having the application running in a WebSphere Application Server cluster using DB2 setup in HADR mode, all on the same rack as shown in Figure 7-2.

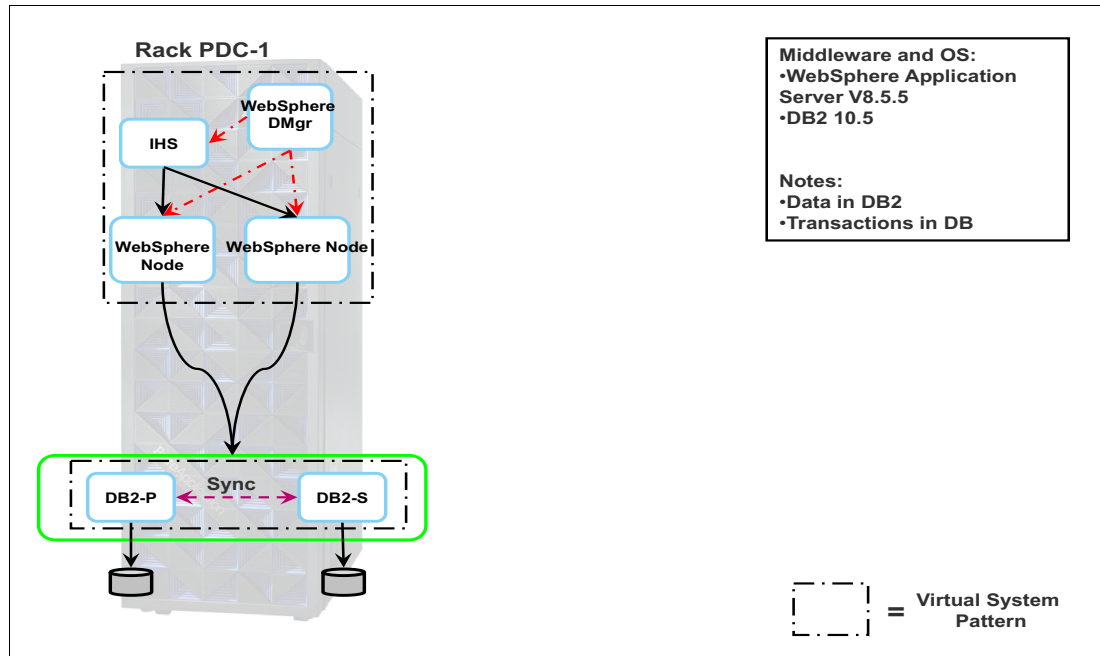


Figure 7-2 Overview of scenario WDB_1

7.2.1 Build the WebSphere Application Server cluster pattern

For instructions about this process, see “Build the WebSphere Application Server cluster pattern” on page 266.

The pattern name for this scenario is *RB HADR WS 1B*.

7.2.2 Deploy WebSphere Application Server cluster pattern

For instructions about this process, see “Deploy WebSphere Application Server cluster pattern: Single rack” on page 269.

The pattern name for this scenario is *RB HADR WS 1B*.

7.2.3 Create a WebSphere Application Server cluster

For instructions about this process, see “Create WebSphere Application Server cluster” on page 273.

7.2.4 Install DB2 JDBC driver

Before creating a data source to connect the sample application to DB2, install the client DB2 database driver. This ensures that the application can connect to a database and run DB2 commands.

For instructions about this process, see “Configure database connectivity for BankTransaction application” on page 273.

7.2.5 Create a JDBC Provider

The application uses a JDBC provider that provides the actual implementation of the JDBC driver for access to DB2.

For instructions about this process, see “Configure database connectivity for BankTransaction application” on page 273. The JDBC provider is scoped to the cell.

Note: The non-XA JDBC Provider needs to be created because WebSphere Application Server requires that the data source used for pointing to the tranlog database needs to be a non-XA one.

7.2.6 Create a J2C alias

Because the database has security enabled, it requires a user ID and password for the connection. To provide these requirements, you must create an authentication resource by creating a J2C alias.

For instructions about this process, see “Create a J2C alias” on page 278.

7.2.7 Create data sources

Create the following data sources scope at the cell level (CHECKING, SAVINGS, and transaction data source).

For instructions about this process, see “Create data sources for SAVINGS, CHECKING, and transaction data source” on page 280.

7.2.8 Install BankTransaction application

Now that you have configured the resources necessary for database connectivity, the application can be installed.

For instructions about this process, see Appendix B: “Install BankTransaction application” on page 284.

7.2.9 Validate the functionality of the application

The installation and configuration of the application must be validated. For instructions about this process, see “Validate BankTransaction application” on page 284.

7.3 Scenario WDB_2: WebSphere Application Server cluster with DB2 HADR, split across two racks with WebSphere transactions stored in database

This scenario is similar to the scenario documented in 7.2, “Scenario WDB_1: WebSphere Application Server cluster and DB2 HADR deployed on a single rack with transactions stored in database” on page 207. The significant difference in this case is that the pattern is split across both racks. As part of the deployment of the WebSphere Application Server pattern, the pattern is deployed across the two racks in the same data center as shown in Figure 7-3.

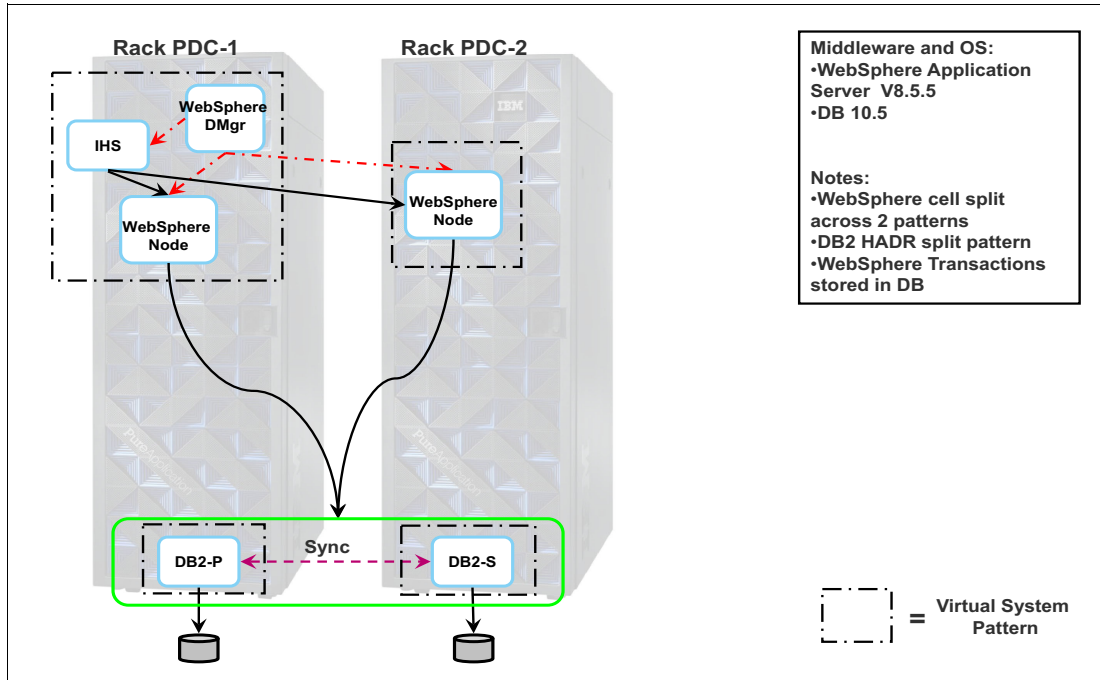


Figure 7-3 Deploying across two racks same data center

The pattern built in the prior scenario does not need to be changed. You can use the same pattern.

This chapter assumes that you have set up the two PureApplication Systems in the PDC to support multi-rack deployment as described in 4.8, “Multisystem environment deployment” on page 71. For this deployment, the Environment Profile RedBooks *HADR External*, which supports multi-rack deployments, was used. It is also assumed that the databases used by the Bank Transaction application have already been deployed using DB2 HADR OLTP Pattern with the primary and secondary databases on the two different racks as shown in Figure 7-4.

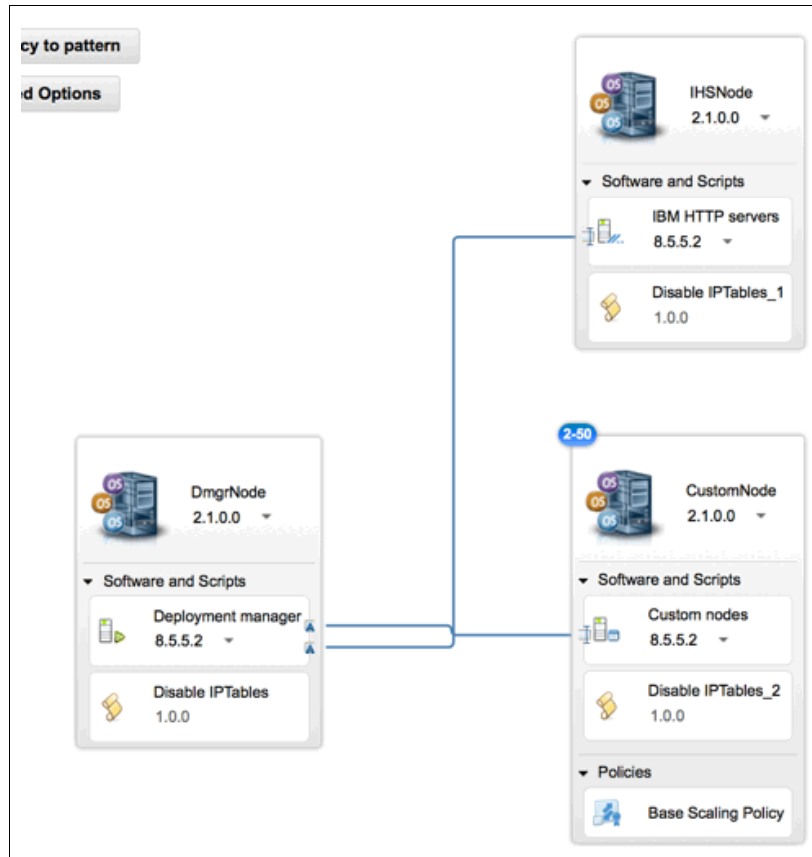


Figure 7-4 Deployment, primary and secondary

To perform the deployment, complete these steps:

1. From the Workload Console, select **Patterns** → **Virtual System Patterns** and look for the RB HADR WAS 1B pattern. If the pattern has not been built, follow the directions in “Build the WebSphere Application Server cluster pattern” on page 266.
2. Deploy the pattern by using the instructions in “Deploy WebSphere Application Server cluster pattern: Multiple rack” on page 270.
3. Create the resources for clustering and database connectivity (if not created in the previous scenario) using the directions in these sections:
 - a. “Create WebSphere Application Server cluster” on page 273
 - b. “DB2 JDBC Database driver installation” on page 274.
 - c. “Create a JDBC Provider” on page 275
 - d. “Create a J2C alias” on page 278
 - e. “Create data sources for SAVINGS, CHECKING, and transaction data source” on page 280

4. Install the application if not already installed from the previous scenario. Follow the instructions in “Install BankTransaction application” on page 284.
5. Validate the scenario by following the instructions in “Validate BankTransaction application” on page 284.

7.4 Scenario WDB_3: Identical WebSphere Application Server cell and DB2 HADR replicated across DR site, with WebSphere transactions stored in database

This scenario sets up a WebSphere Application Server Active/Passive Cell using transactions that are stored in a database. The databases are also configured in an active/passive manner using DB2 HADR. For WebSphere Application Server, you create identical cells on both PDC and SDC as shown in Figure 7-5.

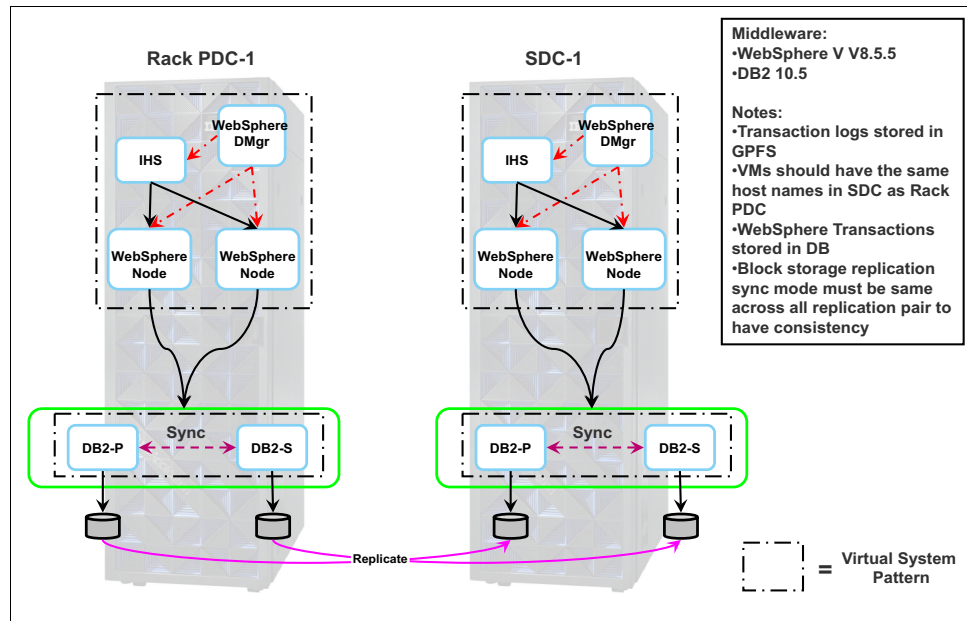


Figure 7-5 Scenario WDB_3

Creating the setup for this scenario includes the following steps:

1. Create a DB2 HADR Pattern.
2. Deploy the pattern on both PDC and SDC.
3. Enable block storage replication between PDC and SDC.
4. Create a WebSphere Application Server Pattern.
5. Deploy the same pattern across both PDC and SDC.

Create the DB2 HADR Pattern

Although the detailed steps for setting up the DB2 HADR pattern are explained in Chapter 5, the important high-level steps are listed here. Use the steps that are outlined in 5.3.3, “Optional: Multiple HADR databases” on page 99 to create a DB2 HADR pattern, which is shown in Figure 7-6.

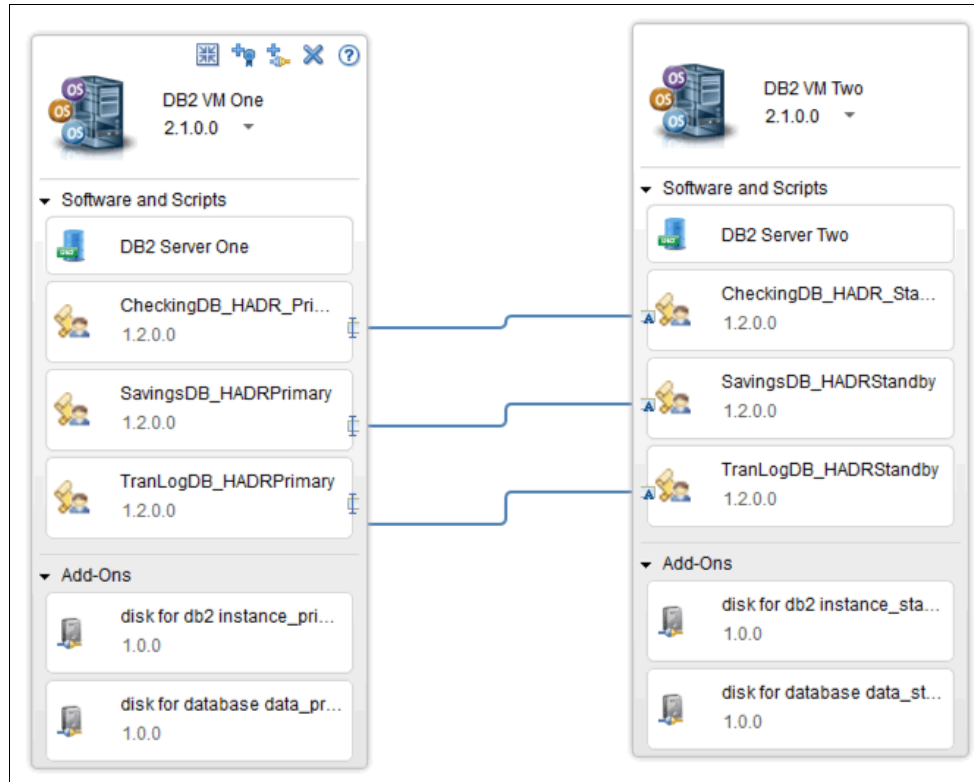


Figure 7-6 DB2 HADR pattern

Deploy the DB2 HADR pattern on both PDC and SDC

The following are the steps to deploy the DB2 HADR pattern on the PDC and SDC:

1. Use the steps that are outlined in 5.6, “Scenario DB2_3: Two systems using block storage replication” on page 119 to customize the pattern to use block storage and deploy the patterns on both PDC and SDC racks.
2. Use the steps that are outlined in 5.6.4, “Enable Block Storage replication” on page 132:
 - a. Stop DB2 on the secondary VM.
 - b. Stop DB2 on the primary VM.
 - c. Detach block storage from each of the VMs.
 - d. Configure replication from the PDC rack using the **Block Storage Replication** menu.
 - e. Accept the request for replication using the **Block Storage Replication** menu on the SDC rack.

Block Storage Replication should now be established between PDC and SDC.

Create the WebSphere Application Server Pattern

The following are the steps to create the WebSphere Application Server pattern:

1. From the Workload Console, select **Patterns** → **Virtual System Patterns** and look for the RB_HADR_WAS_3_Existing_DB2HADR pattern. If the pattern has not been built, follow the directions in Appendix B, “Common WebSphere Application Server configuration tasks” on page 265. The pattern should resemble Figure 7-7.

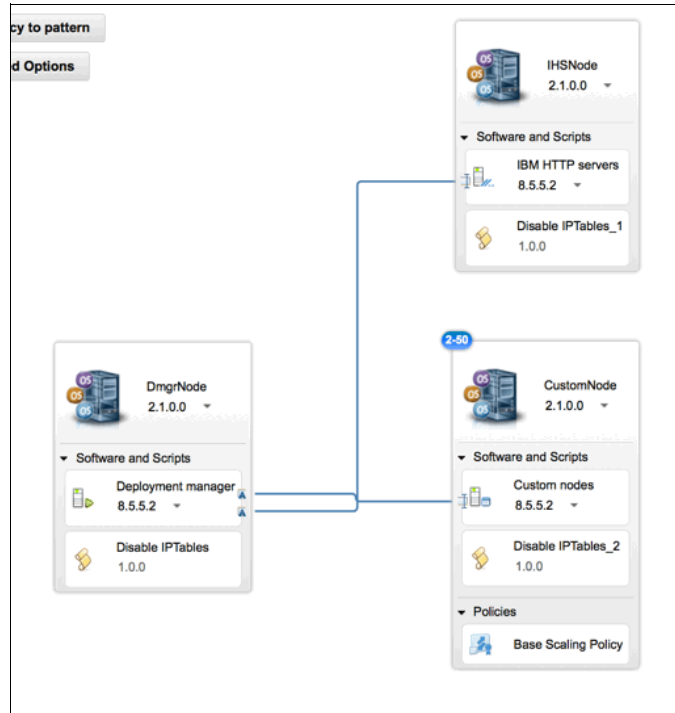


Figure 7-7 WebSphere pattern with DMgr, custom nodes, and IBM HTTP Server

2. Export the pattern:
 - a. Click **Export**.
 - b. Enter file name that you can recognize because this pattern will be imported onto the SDC rack (see Figure 7-8).



Figure 7-8 Export a pattern

3. Import the pattern on the SDC rack:
 - a. Log in to the SDC rack.
 - b. From the Workload console, select **Patterns** → **Virtual System**.

- c. Click **Import** as shown in Figure 7-9.



Figure 7-9 Import option

- d. Browse to the archive file created when you exported the pattern from PDC and click **OK** as shown in Figure 7-10.

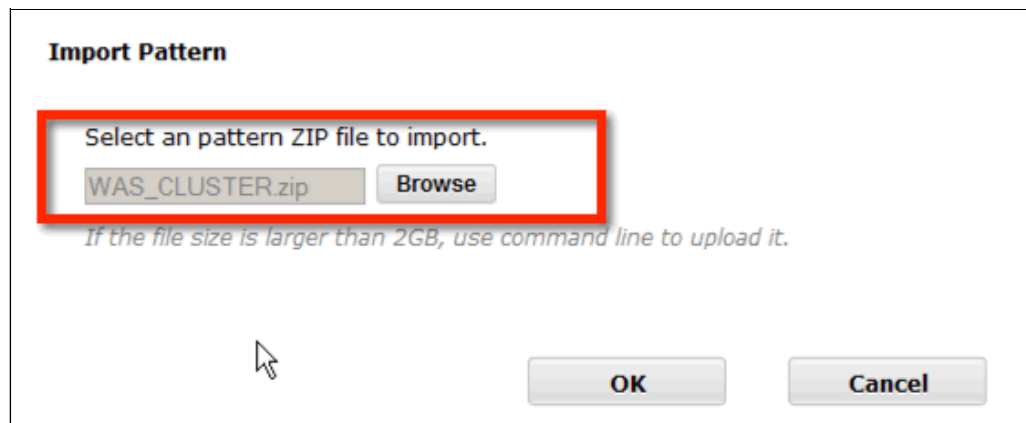


Figure 7-10 Exported pattern

After being imported, the same patterns exist on both the PDC and SDC.

Deploy the WebSphere Application Server Pattern

The following are the steps to deploy the WebSphere Application Server pattern:

1. On the PDC, deploy the WebSphere Application Server pattern that was created in “Create the WebSphere Application Server Pattern” on page 213. Follow the instructions in “Deploy WebSphere Application Server cluster pattern: Single rack” on page 269.
2. During deployment, an environment profile, that was created for the DR scenarios was used. The details of the environment profile are documented in 4.7, “Network configuration and cloud resources configuration” on page 65. See Figure 7-11.

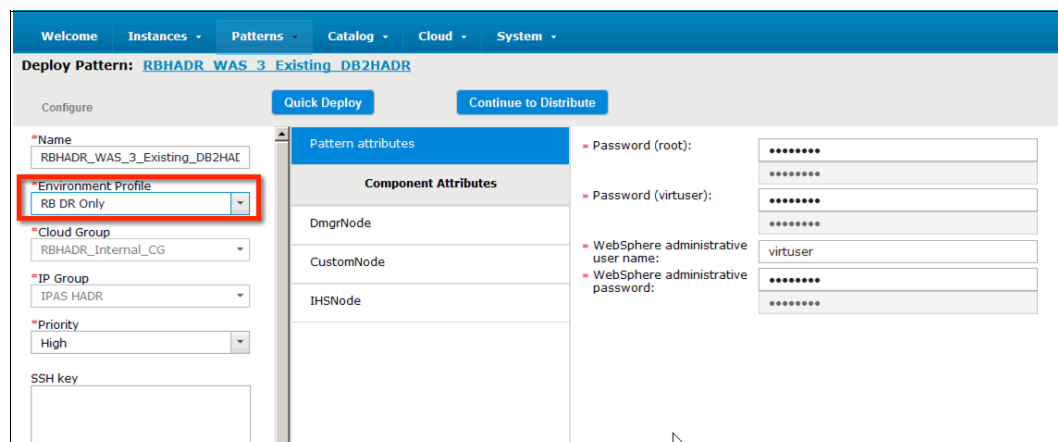


Figure 7-11 Environment profile selected for deployment

3. Make note of the host names of each of the VMs that are part of your cell. You can find the host name of a VM by expanding the virtual machine on the instances window as shown in Figure 7-12.

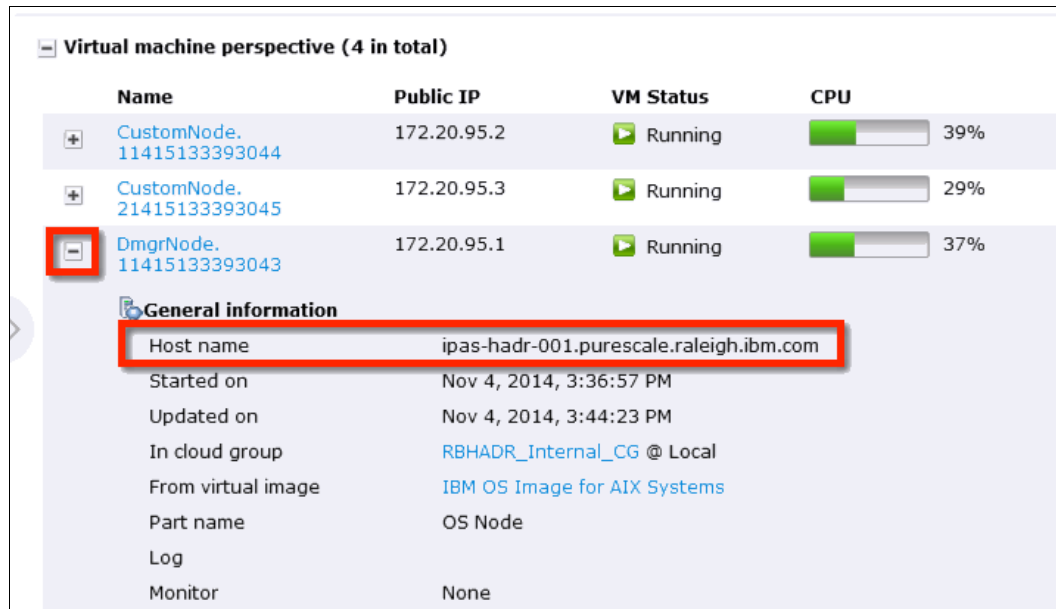


Figure 7-12 Host name of Deployment Manager

Note: The host names the pattern instance in used in this publication were:

- ▶ Deployment Manager: ipas-hadr-001.purescale.raleigh.ibm.com; IP: 172.20.95.1
- ▶ Custom Node 1: ipas-hadr-002.purescale.raleigh.ibm.com; IP: 172.20.95.2
- ▶ Custom Node 2: ipas-hadr-003.purescale.raleigh.ibm.com; IP: 172.20.95.3
- ▶ IHS Node: ipas-hadr-040.purescale.raleigh.ibm.com; IP: 172.20.95.40

4. Deploy the pattern that you imported into SDC. Follow the instructions in “Deploy WebSphere Application Server cluster pattern: Single rack” on page 269. When deploying into SDC, use an environment profile that requires IP addresses to be specified at deployment time. The cell is passive and is only activated in a DR situation. Because of this requirement, the cell must have the same host names as the cell on the active server. See Figure 7-13. Click **Continue to Distribute**. This action allows you to specify IP addresses for each part.

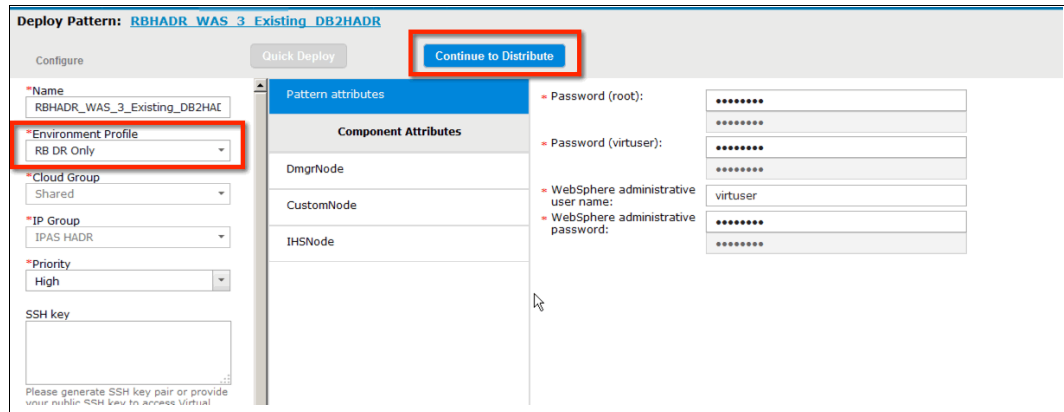


Figure 7-13 Environment profile for specifying IP addresses

5. After the pattern is deployed, notice that although the IP address is different, the host names that are assigned to each VM are the same as those assigned on the PDC, as shown in Figure 7-14.

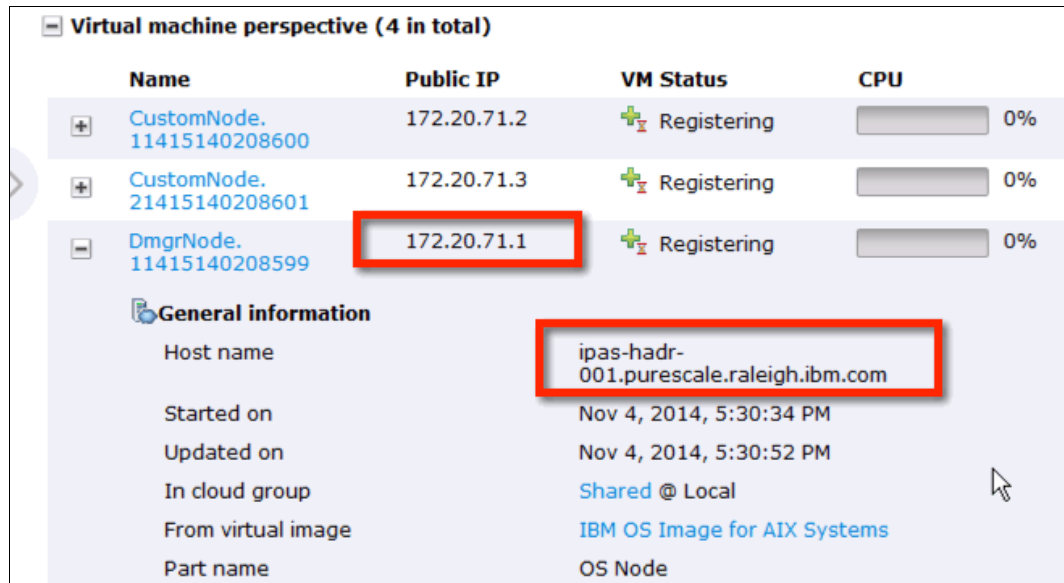


Figure 7-14 Host name and IP address

Validate the test case

Use the following steps to validate the test case:

1. Set up the application using the administrative console on the PDC.
 - a. Create a WebSphere Application Server cluster called MyCluster, with a cluster member on each custom node. Follow the instructions in “Create WebSphere Application Server cluster” on page 273. See Figure 7-15 for an example.

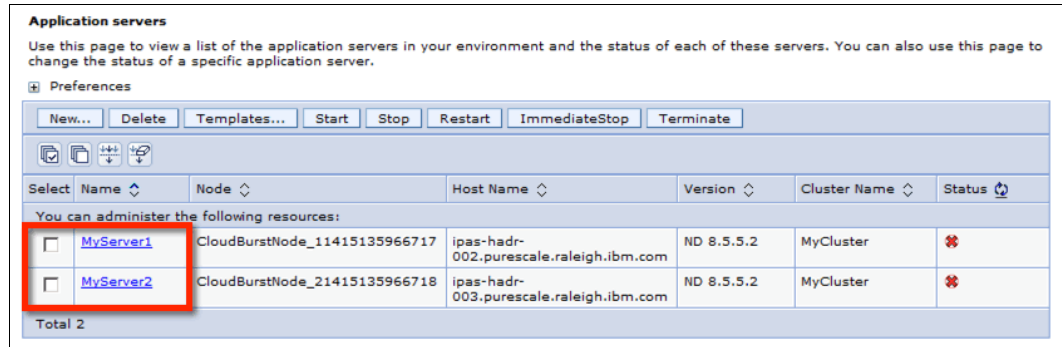


Figure 7-15 WebSphere Application Server cluster configuration

- b. Create two JDBC drivers. The application needs an XA JDBC Driver and the tranlog requires a non-XA JDBC Driver. Use the instructions that are provided in “DB2 JDBC Database driver installation” on page 274. Figure 7-16 shows a sample.

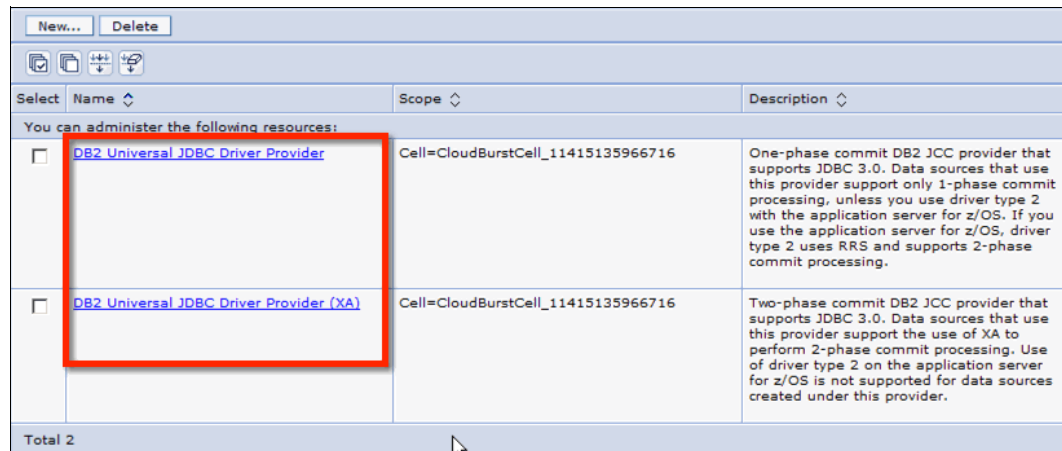


Figure 7-16 JDBC drivers

2. Create data sources according to the values in Table 7-1. The J2C Aliases must have the same names on the PDC and SDC racks. You can use the instructions in “Create data sources for SAVINGS, CHECKING, and transaction data source” on page 280 as a guide.

Table 7-1 Data sources for test case validation

| Data Source | JNDI | JDBC Driver Type | DB Server | J2C Alias |
|-------------|---------------|------------------|-------------|--|
| CHECKING | jdbc/checking | XA | 172.20.88.3 | CloudBurstNode_11415135966716/CHECKING |

| Data Source | JNDI | JDBC Driver Type | DB Server | J2C Alias |
|-------------------------|--------------|------------------|-------------|--|
| SAVINGS | jdbc/savings | XA | 172.20.88.3 | CloudBurstNode_11415135966716/CHECKING |
| Transaction data source | jdbc/trands | NON-XA | 172.20.88.3 | CloudBurstNode_11415135966716/CHECKING |

- c. Install the application named **BankTransaction**. Instructions are found in “Install BankTransaction application” on page 284.
3. For each server under the Transaction Service, configure the Transaction log Directory. See Figure 7-17.

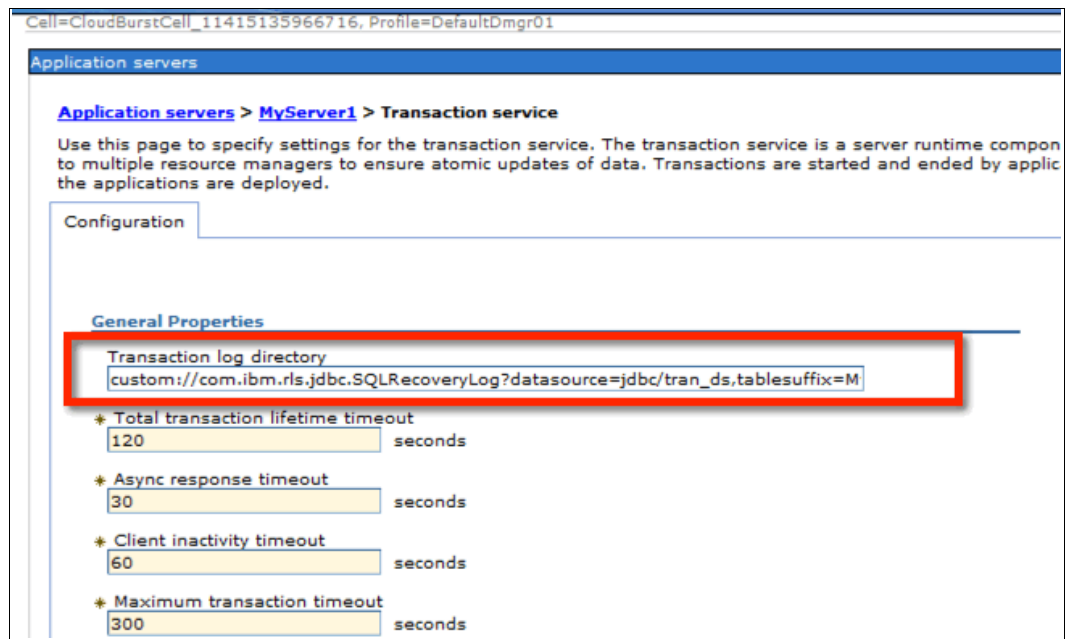


Figure 7-17 Transaction Service configuration

Repeat these steps on SDC. Follow the instructions in these sections:

1. “Create the DB2 HADR Pattern” on page 212
2. “Deploy the DB2 HADR pattern on both PDC and SDC” on page 212
3. “Create the WebSphere Application Server Pattern” on page 213
4. “Deploy the WebSphere Application Server Pattern” on page 214

However, there are a few differences:

1. Create the data sources (see “Create data sources for SAVINGS, CHECKING, and transaction data source” on page 280) and use the same J2C Alias as on the cell in PDC. However, the database should point to the primary DB2 VM deployed as part of the DB2 HADR pattern on SDC. See Table 7-2 for more details.

Table 7-2 Data source needed on the SDC

| Data Source | JNDI | JDBC Driver Type | DB Server | J2C Alias |
|-------------------------|---------------|------------------|-------------|--|
| CHECKING | jdbc/checking | XA | 172.20.64.3 | CloudBurstNode_11415135966716/CHECKING |
| SAVINGS | jdbc/savings | XA | 172.20.64.3 | CloudBurstNode_11415135966716/CHECKING |
| Transaction data source | jdbc/trands | NON-XA | 172.2064.3 | CloudBurstNode_11415135966716/CHECKING |

2. There is a requirement to have the J2CAliases the same as defined in PDC. Use **wsadmin** to accomplish this task. Ideally, this step should be scripted. The commands for setting the J2C Alias are shown in Figure 7-18.

```
wsadmin>security = AdminConfig.getid('/Cell:CloudBurstCell_11415140208599/Security:/')
wsadmin>print security
(cells/CloudBurstCell_11415140208599|security.xml#Security_1)
wsadmin>alias = ['alias', 'CloudBurstNode_11415135966716/CHECKING']
wsadmin>
wsadmin>userid = ['userid', 'db2inst1']
wsadmin>password = ['password', 'passw0rd']
wsadmin>jassattrs = [alias,userid,password]
wsadmin>print AdminConfig.create('JAASAuthData', security, jassattrs)
(cells/CloudBurstCell_11415140208599|security.xml#JAASAuthData_1415151868573)
wsadmin>AdminConfig.save()
''
wsadmin>alias = ['alias', 'CloudBurstNode_11415135966716/SAVINGS']
wsadmin>userid = ['userid', 'db2inst1']
wsadmin>password = ['password', 'passw0rd']
wsadmin>jassattrs = [alias,userid,password]
wsadmin>print AdminConfig.create('JAASAuthData', security, jassattrs)
(cells/CloudBurstCell_11415140208599|security.xml#JAASAuthData_1415151952306)
wsadmin>AdminConfig.save()
''
wsadmin>alias = ['alias', 'CloudBurstNode_11415135966716/TRANLOG']
wsadmin>jassattrs = [alias,userid,password]
wsadmin>print AdminConfig.create('JAASAuthData', security, jassattrs)
(cells/CloudBurstCell_11415140208599|security.xml#JAASAuthData_1415152008758)
wsadmin>AdminConfig.save()
''
wsadmin>
```

Figure 7-18 Commands to set J2C Aliases

3. Install the application named **BankTransaction** by following the instructions on “Install BankTransaction application” on page 284.

Set the Transaction Service for each Application Server in the cluster. Follow the instructions at the following website to set up the appserver to use a database for transaction recovery:

<http://www-01.ibm.com/support/docview.wss?uid=swg27038432>

4. Start both application servers.

5. Test the application on PDC and ensure that transaction recovery is working. Use the instructions in “Validate BankTransaction application” on page 284. The results shows that the application is working as designed and the failure of one cluster member does not cause a lost transaction.
6. Validate the transaction recovery on SDC:
 - a. Remember the amounts above. Transfer \$100.00 from the Savings to the Checking account.
 - b. While the application is waiting, shut down the various components in the following order:
 - Application Server JVMs. Use `ki11 -9` because that is close to a unrecoverable failure.
 - DB2 Secondary (follow the steps that are outlined in “Stop DB2 on the Virtual System Instance in the PDC” on page 142.)
 - DB2 Primary (follow the steps that are outlined in “Stop DB2 on the Virtual System Instance in the PDC” on page 142.)
 - c. Detach the block storage volumes that are currently attached to the VMs in PDC. Follow the steps that are outlined in “Detach Block Storage volumes from the VMs in the PDC” on page 143. You should see a status similar to the status shown in Figure 7-19.

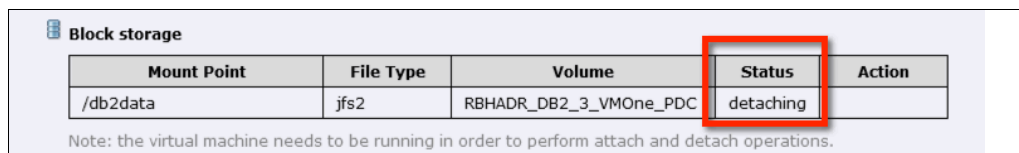


Figure 7-19 Detaching a block storage volume

7. Perform a failover of the storage volumes from PDC to SDC.
 - a. From the system console on the PDC rack, go to **System** → **Block Storage Replication Profiles**.
 - b. Look for the volumes configured for replication. Click **Failover** as shown in Figure 7-20.

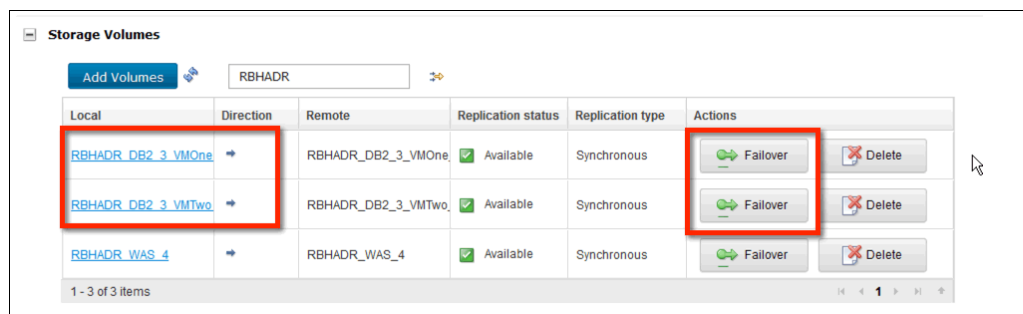


Figure 7-20 Failover

Note: If this was an unplanned failover, such as a rack failure, this option is not possible. In that case, go to the same window on the other rack and click **Failover**.

- Attach the volumes to the DB2 VMs in the SDC as shown in Figure 7-21.

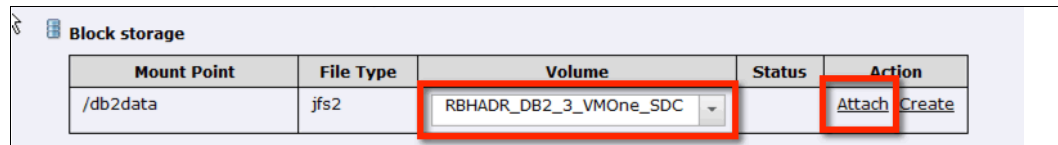


Figure 7-21 Attaching volumes to the DB2 VMs

- Log in to each of the DB2 VMs as **root**.
- Start DB2 using the **chrg** command as documented in “Start DB2 on the Virtual System Instance in the SDC” on page 148. If DB2 does not start, use the **db2start** command as the instance owner.
- For each VM, update the database configuration to show the correct local and remote hosts as shown in Figure 7-22. For more information, see “Manually updating the DB2 HADR configuration on the VMs” on page 151.

```

$ hostname
ipas-pvm-064-025.purescale.raleigh.ibm.com
$ db2 get db cfg for savings | grep "host name"
HADR local host name          (HADR_LOCAL_HOST) = ipas-pvm-088-004
HADR remote host name        (HADR_REMOTE_HOST) = ipas-pvm-088-003
$ db2 update db cfg for savings using HADR_LOCAL_HOST ipas-pvm-064-025
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$se
ksh: ^[se: not found.
$ set -o vi
$ db2 update db cfg for checking using HADR_LOCAL_HOST ipas-pvm-064-025
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for tranlog using HADR_LOCAL_HOST ipas-pvm-064-025
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for savings using HADR_REMOTE_HOST ipas-pvm-064-023
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for checking using HADR_REMOTE_HOST ipas-pvm-064-023
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for tranlog using HADR_REMOTE_HOST ipas-pvm-064-023
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
$
$
$
$
$ db2 get db cfg for tranlog | grep "host name"
HADR local host name          (HADR_LOCAL_HOST) = ipas-pvm-064-025
HADR remote host name        (HADR_REMOTE_HOST) = ipas-pvm-064-023
$

```

Figure 7-22 Local and remote host names

- Stop and restart the database instances in both VMs using the **chrg** command.
- Because the databases are recovered, start the application server named **MyServer1** in SDC as shown in Figure 7-23.

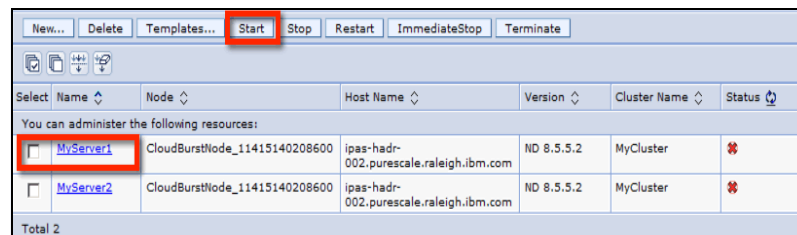


Figure 7-23 Starting an application server

14. Start the BankTransaction application from a browser. Notice that the balance is showing the correct values (see Figure 7-24).

| Savings account balance | | |
|-------------------------|------|----------|
| ACCOUNT_NUMBER | NAME | AMOUNT |
| 12345 | IPAS | 19800.00 |

| Checking account balance | | |
|--------------------------|------|---------|
| ACCOUNT_NUMBER | NAME | AMOUNT |
| 54321 | IPAS | 8200.00 |

Figure 7-24 Saving accounts balances

Note: Not all modes of database recovery are shown in this chapter because doing so is beyond the scope of this chapter. For details about planned and unplanned failovers of the database, see 5.6.6, “Planned Failover to SDC” on page 142 and 5.6.7, “Unplanned Failover to SDC” on page 156.



High availability and disaster recovery scenarios for WebSphere MQ

This chapter describes several High Availability and Disaster Recovery (HADR) scenarios around WebSphere MQ patterns in PureApplication System. There are several high availability disaster recovery (HADR) options possible with WebSphere MQ such as MQ Clustering and Multi-Instance Queue Manager (MIQM).

The scenarios in this chapter focus on the use of MIQM. With MIQM, multiple instances of the same queue manager are configured on multiple systems, and each of those instances uses a network storage shared file system such as NFS, GPFS, and so on, for the queue manager data, logs, and more. MIQM provides high availability of WebSphere MQ Queue Managers without using any High Availability Coordinator.

The MIQM has an active server and one or more standby servers that host the same queue manager and the shared file system. When the active server fails, there is an automatic switch to the standby server. The active server (queue manager) instance accepts requests from applications or other queue managers. The lock on the queue manager data on the shared file system is held by the active instance, while the standby server does periodic checks to see whether the active queue manager is still active. The standby server instance acquires the lock if the active queue manager fails. The standby becomes the new active server.

The messaging provider in WebSphere Application Server (starting with V7.0.13) can connect to a MIQM, specified through the WebSphere MQ messaging provider connection factory or activation specification custom properties. Other WebSphere Java Message Service (JMS) clients can use the automatic client reconnect to connect to MIQM.

For the network storage shared file system, you can use NFS, but PureApplication System offers a better file sharing solution in GPFS.

The following link provides more details about how WebSphere MQIM works:

https://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wmq_v7/wmq/7.0.1/Details/iea_701_120_multi_instancei.pdf?dmuid=20091218113354709936

In this chapter, several common WebSphere MQ patterns are shown to provide HADR across single or multiple PureApplication System systems, as shown in Table 8-1.

Table 8-1 Overview of the WebSphere MQ scenarios.

| Scenario | PDC | SDC | HA | DR |
|-----------------|-----------------------------|------------|-----------|-----------|
| WMQ_1 | Yes | No | Yes | No |
| WMQ_2 | Yes - across 2 racks in PDC | No | Yes | No |
| WMQ_3 | Yes | Yes | Yes | Yes |

The prerequisites for creating a WebSphere MQ MIQM topology is GPFS setup, which is described in Chapter 4, “Infrastructure setup” on page 47.

The details of these scenarios are discussed in this chapter. You can also see details of the WebSphere MQ scenarios in 3.5, “HADR scenarios for WebSphere MQ” on page 43.

The following topics are covered in this chapter:

- ▶ Common assets used in WebSphere MQ scenarios
- ▶ Scenario WMQ_1: WebSphere MQ Primary and standby in the same pattern deployed on a single rack
- ▶ Scenario WMQ_2: WebSphere MQ primary and standby in different patterns deployed on two different racks in the same data center
- ▶ Scenario WMQ_3: WebSphere MQ primary and standby in the different patterns deployed on two different racks across different data centers

8.1 Common assets used in WebSphere MQ scenarios

This section shows all the assets that are used to build the patterns that make up the solution for the different scenarios. The assets include the WebSphere MQ images, the add-on policies for GPFS, and the script packages. The details of each of those assets are described below.

8.1.1 Image Parts

The **WebSphere MQ 7.5.0.x** image is used for primary and standby QManagers. The image should be imported in the Workload catalog for virtual images in the pattern editor, as shown in Figure 8-1.

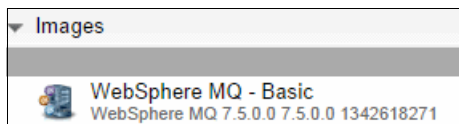
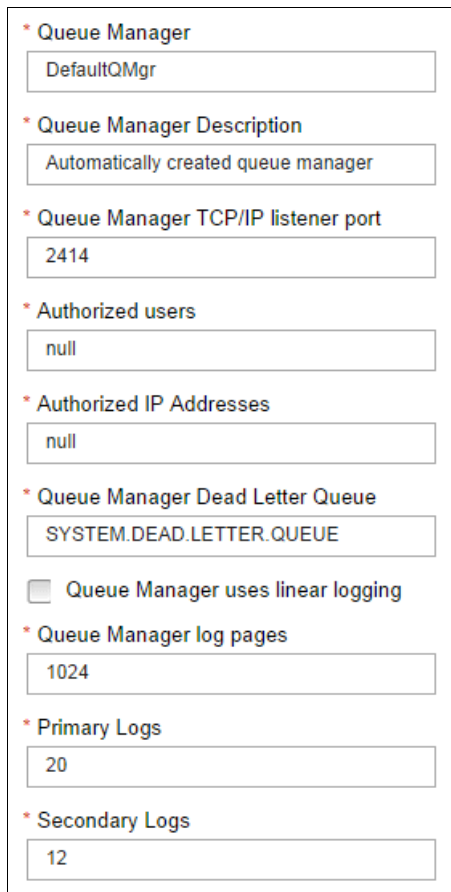


Figure 8-1 WebSphere MQ 7.5.0.x image

The key parameters of the WebSphere MQ image are shown in Figure 8-2. It creates a default QManager, which is deleted by using a script package that is executed at deployment time.

A screenshot of a configuration form for a WebSphere MQ image. The form contains several fields, each with a label and a value:

- * Queue Manager: DefaultQMgr
- * Queue Manager Description: Automatically created queue manager
- * Queue Manager TCP/IP listener port: 2414
- * Authorized users: null
- * Authorized IP Addresses: null
- * Queue Manager Dead Letter Queue: SYSTEM.DEAD.LETTER.QUEUE
- Queue Manager uses linear logging
- * Queue Manager log pages: 1024
- * Primary Logs: 20
- * Secondary Logs: 12

Figure 8-2 Parameters for default QManager

8.1.2 Policies

For all the scenarios, dedicated QManagers were created that use a shared directory provided by GPFS.

The **GPFS Client Policy** add-on creates or uses an existing file set on the GPFS server file system. The directories in the file set are linked to a local directory within the WebSphere MQ VM. The locally linked file directory is then used for QManager data, logs, and errors. The GPFS policy parameters that are used in the scenario are as follows:

- ▶ **GPFS File System Information:** RBHADRfileSystem was used. This file system was created when the GPFS server environment was created. The same file system is used for all scenarios in this book.
- ▶ **GPFS file set directory:** The scenario name is used as the unique file set directory, such as WMQ_1, WMQ_2, and WMQ_3.
- ▶ **Storage Maximum Limit:** 100M (MB) was used for this file set.
- ▶ **Directory to link on local system:** /opt/MQShare. This is the local link directory on the virtual machine that is specified when creating new QManagers.

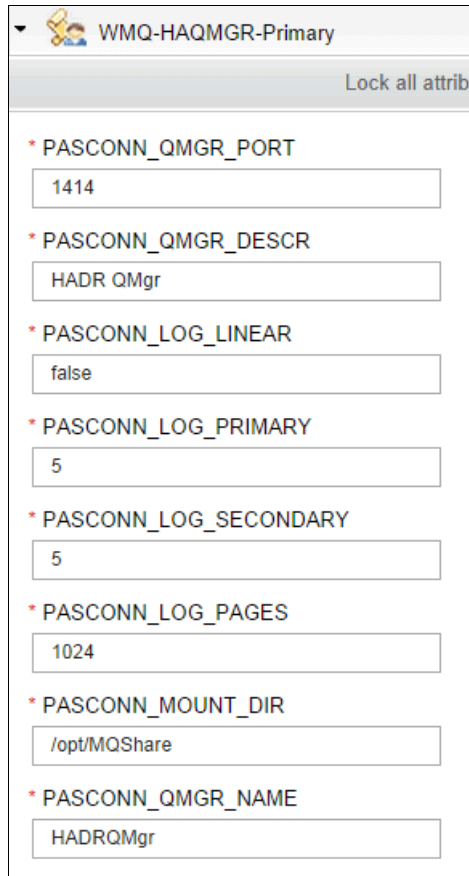
8.1.3 Script packages and parameters used in the pattern

The following script packages are available to download. For details about how to download these script packages for use, see Appendix C, “Additional material” on page 287.

WMQ-De1QMgr: This script deletes the default QManager that is created as part of WebSphere MQ instance deployment. This script does not require any parameters and is run at deployment time.

WMQ-HAQMGR-Primary: This script creates a primary QManager using the specified directory for data and logs. The directory is the locally linked directory to the GPFS file set directory created for this scenario. This script is run “on-demand” after the pattern has been deployed. The key parameters are described below and shown in Figure 8-3 on page 227.

- ▶ **PASCONN_QMGR_PORT:** Unique port for the QManager. Port 1414 was used for this scenario.
- ▶ **PASCONN_QMGR_NAME:** Name of the QManager. HADRQMgr was used.
- ▶ **PASCONN_MOUNT_DIR:** The directory where the QManager data and logs are created. /opt/MQShare was used and was linked to the GPFS file set directory specified in the GPFS client policy.

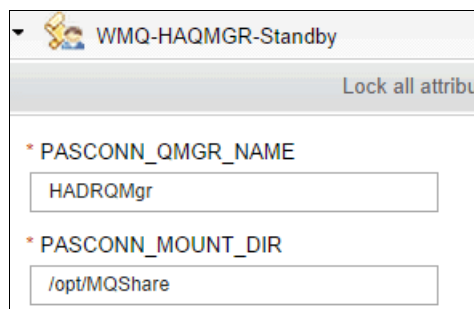


| Parameter | Value |
|-------------------------|--------------|
| * PASCONN_QMGR_PORT | 1414 |
| * PASCONN_QMGR_DESCR | HADR QMgr |
| * PASCONN_LOG_LINEAR | false |
| * PASCONN_LOG_PRIMARY | 5 |
| * PASCONN_LOG_SECONDARY | 5 |
| * PASCONN_LOG_PAGES | 1024 |
| * PASCONN_MOUNT_DIR | /opt/MQShare |
| * PASCONN_QMGR_NAME | HADRQMGr |

Figure 8-3 WMQ-HAQMGR-Primary parameters

WMQ-HAQMGR-Standby: This script creates a standby QMgr using the specified directory for data and logs. The directory used is the locally linked directory to the GPFS file set directory created for this scenario. This script is run “on-demand” after the pattern has been deployed. The key parameters are described below and shown in Figure 8-4:

- ▶ **PASCONN_QMGR_NAME:** Name of the QManager. This name should be the same as the primary QManager, HADRQMGr.
- ▶ **PASCONN_MOUNT_DIR:** The directory where the QManager data and logs are created by the primary QManager. /opt/MQShare was used in this scenario. This directory is linked to the GPFS file set directory specified in the GPFS client policy that is created by the primary QManager instance.



| Parameter | Value |
|---------------------|--------------|
| * PASCONN_QMGR_NAME | HADRQMGr |
| * PASCONN_MOUNT_DIR | /opt/MQShare |

Figure 8-4 WMQ-HAQMGR-Standby parameters

WebSphere MQ Run MQSC Command: This script runs a single MQSC command on the QManager. For example, you can create the Queue on the QManager using this script. This script is run “on-demand” after the pattern has been deployed. The key parameters are as follows, and are shown in Figure 8-5:

- ▶ QMGR_NAME: Name of the QManager, HADRQMgr.
- ▶ MQSC_COMMAND: Provide the MQSC command to run.

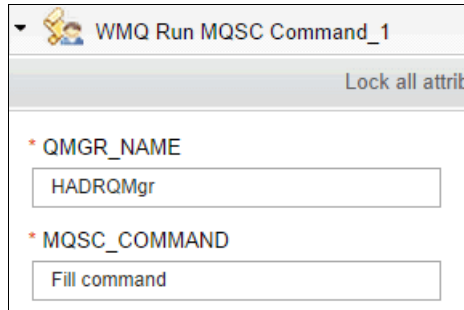


Figure 8-5 WebSphere MQ Run WMQSC Command parameters

WebSphere MQ Run MQSC Scripts: This script runs multiple MQSC commands from a file that is packaged with the script compressed file. It is available for you to use. The “WebSphere MQ Run MQSC Command” was chosen because the authors had few commands to run. This script is configure to run “on-demand”.

8.1.4 Prerequisites

This scenario requires the following prerequisites:

- ▶ A deployed GPFS environment.
- ▶ An instance GPFS primary and optional mirror (for HA of the shared file system) servers are deployed.
- ▶ IBM Shared Service for GPFS is deployed in the same cloud group where the pattern is deployed. Based on the scenario, the GPFS shared service connects to either GPFS Primary-Mirror or GPFS Primary-Passive configuration based on the scenario. The usage is described in details along with the scenario.

8.2 Scenario WMQ_1: WebSphere MQ Primary and standby in the same pattern deployed on a single rack

This scenario consists of two WebSphere MQ instances, one being the primary (active) and the other being the standby QManager.

The following are the high-level steps of building the pattern, named “RB HADR WMQ_1”, using the Pattern builder:

1. From the Image section of the palette, complete these steps:
 - a. Drag one WebSphere MQ image part to be the primary QManager.
 - b. Name the part WMQ_Primary.

- c. Provide the parameters to create the default QManager. On this WebSphere MQ Primary part, add the following script packages with the parameters described in 8.1, “Common assets used in WebSphere MQ scenarios” on page 225:
 - WMQ-DelQMgr
 - WMQ-HAQMGR-Primary
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts
2. From the Image section of the palette, complete these steps:
 - a. Drag another WebSphere MQ image part to be the standby QManager.
 - b. Name the part WMQ_Standby.
 - c. Provide the parameters to create the default QManager. On this WebSphere MQ Standby part, add the following script packages with the parameters described in 8.1, “Common assets used in WebSphere MQ scenarios” on page 225:
 - WMQ-DelQMgr
 - WMQ-HAQMGR-Standby
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts

The pattern looks like Figure 8-6.

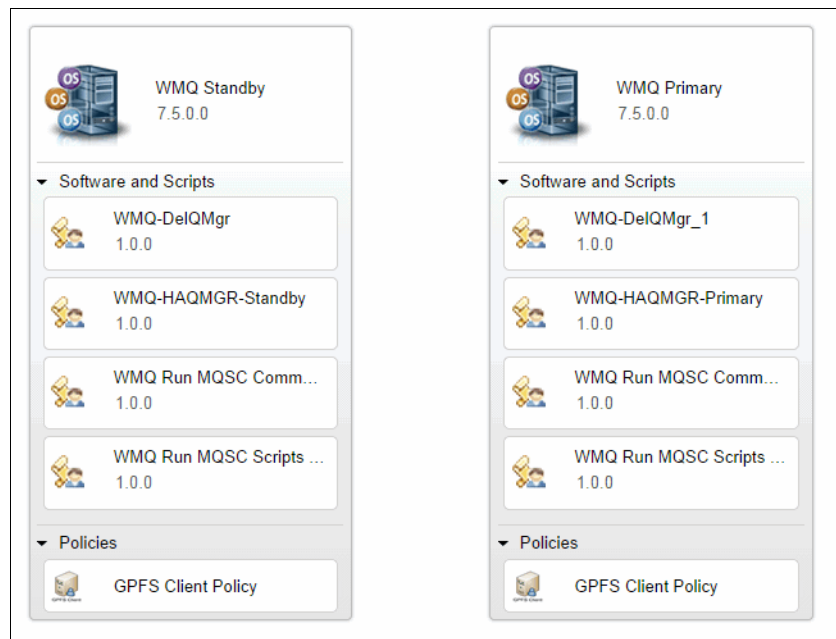
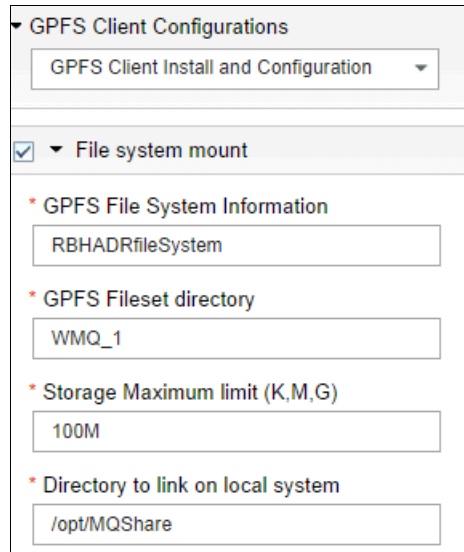


Figure 8-6 WebSphere MQ Primary and Standby

The GPFS client connects through the GPFS shared service to the GPFS Primary-mirror setup as described in 4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53. The GPFS client policy parameters on both the WebSphere MQ parts look like Figure 8-7.



GPFS Client Configurations

GPFS Client Install and Configuration

File system mount

* GPFS File System Information
RBHADRfileSystem

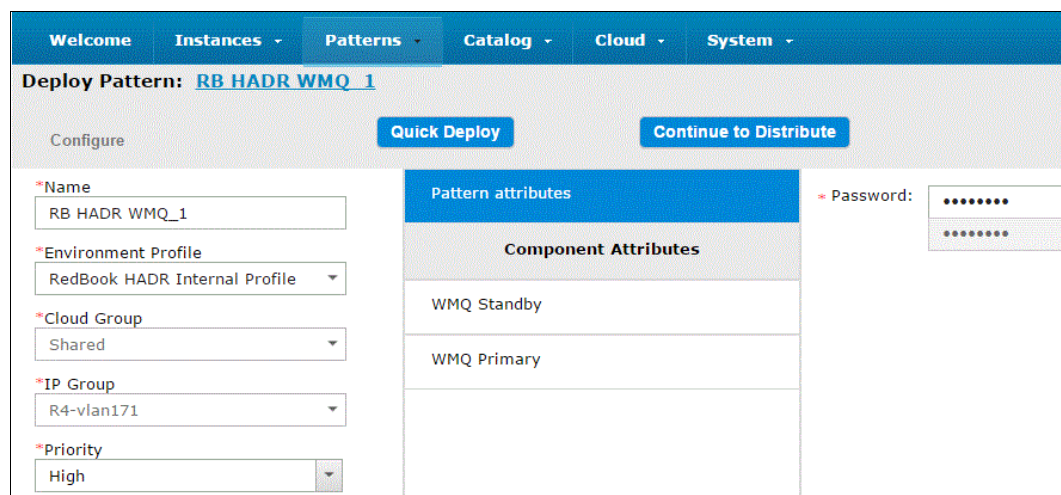
* GPFS Fileset directory
WMQ_1

* Storage Maximum limit (K,M,G)
100M

* Directory to link on local system
/opt/MQShare

Figure 8-7 GPFS client policy

3. The pattern is now ready for deployment on the single rack, as shown in Figure 8-8. Complete these steps:
 - a. From the Workload Console, select **Patterns** → **Virtual System**.
 - b. Select the pattern.
 - c. Click the **Deploy** option for the virtual system pattern to start the deployment.
 - d. Provide the passwords for the VMs. The system creates two VMs (WMQ_Primary and WMQ_Standby).



Welcome Instances Patterns Catalog Cloud System

Deploy Pattern: [RB HADR WMQ_1](#)

Configure **Quick Deploy** **Continue to Distribute**

*Name: RB HADR WMQ_1

*Environment Profile: RedBook HADR Internal Profile

*Cloud Group: Shared

*IP Group: R4-vlan171

*Priority: High

| Pattern attributes | Password: |
|----------------------|-----------|
| Component Attributes | |
| WMQ Standby | |
| WMQ Primary | |

Figure 8-8 Deploying the pattern

4. Create the QManagers and the Queue.

- a. After the two VMs are deployed, run the following “on-demand” scripts that are part of the deployed VM. These scripts must run in the following sequence:
 - i. On WMQ_Primary, run the WMQ-HAQMGR-Primary script with the parameters that are shown in Figure 8-9.

Fill in the required values for this script package

Required parameters value for executing script package

| | |
|-------------------------|--|
| * PASCONN_QMGR_PORT | <input type="text" value="1414"/> |
| * PASCONN_QMGR_DESCR | <input type="text" value="HADR QManager"/> |
| * PASCONN_LOG_LINEAR | <input type="text" value="false"/> |
| * PASCONN_LOG_PRIMARY | <input type="text" value="5"/> |
| * PASCONN_LOG_SECONDARY | <input type="text" value="5"/> |
| * PASCONN_LOG_PAGES | <input type="text" value="1024"/> |
| * PASCONN_MOUNT_DIR | <input type="text" value="/opt/MQShare"/> |
| * PASCONN_QMGR_NAME | <input type="text" value="HADRQMgr"/> |

Product administrator user name and password

User name:

Password:

Figure 8-9 WMQ-HADRQMgr-Primary script

- ii. On WMQ_Standby, run WMQ-HAQMGR-Standby script with the parameters that are shown in Figure 8-10.

Fill in the required values for this script package

Required parameters value for executing script package

* PASCNN_QMGR_NAME

* PASCNN_MOUNT_DIR

Product administrator user name and password

User name:

Password:

Figure 8-10 WMQ-HADRQMGR-Standby script

- iii. On WMQ_Primary, run WebSphere MQ Run MQSC Command script to create a local persistent queue, with the parameters shown in Figure 8-11 (the MQSC command is **Define QLOCAL(HADRQ1) DEFPSIST(YES)**).

Fill in the required values for this script package

Required parameters value for executing script package

* QMGR_NAME

* MQSC_COMMAND

Product administrator user name and password

User name:

Password:

Figure 8-11 RunMQSCCommand script

5. Verify the environment using these steps:
 - a. To send and receive messages, use PuTTY to open two SSH sessions to WMQ_Primary and WMQ_Standby. Use the IP addresses as shown in the virtual instance console for this deployment (See Figure 8-12).

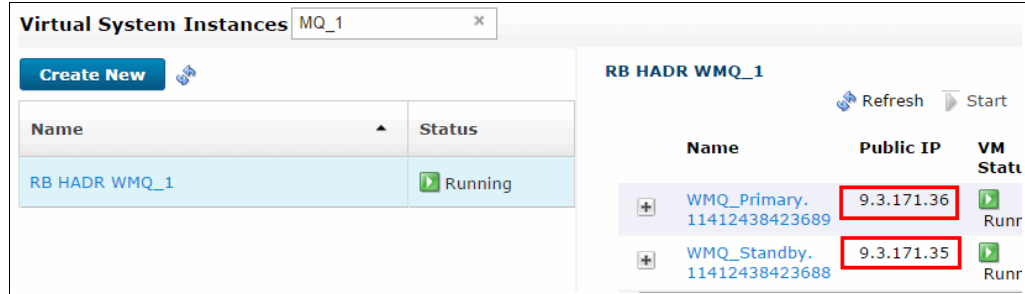


Figure 8-12 WMQ_Primary and WMQ_Standby IP addresses

- b. Check the primary and standby QManagers by running the following commands:
 - Using the WMQ_Primary VM SSH session, check the QManager:


```
su - mqm
```

```
dspmqr (this shows the HADRQMgr running as primary, as shown in the following output)
```

```
QMNAME(HADRQMgr) STATUS (Running)
```
 - Using the WMQ_Standby VM SSH session, check the QManager:


```
su - mqm
```

```
dspmqr (this shows the HADRQMgr running as primary, shown in the following output)
```

```
QMNAME(HADRQMgr)STATUS (Running as standby)
```

- c. On WMQ_Primary VM SSH session, put the test messages as described below and shown in Figure 8-13:

```
cd /opt/mqm/samp/bin
./amqsput HADRQ1 HADRQMGr (this allows you to put the messages to the Queue.)
This is test 1
This is test 2
This is test 3
End
<enter> (this ends the sample)
```

```
login as: root
root@9.3.171.36's password:
Last login: Sun Oct 19 17:05:53 2014 from sig-9-65-4-221.mts.ibm.com
IBM WebSphere MQ Hypervisor Edition for Red Hat Enterprise Linux
[root@ipas-lpar-9-3-171-36 ~]# su - mqm
-bash-4.1$ cd /opt/mqm/samp/bin/
-bash-4.1$ ./amqsput HADRQ1 HADRQMGr
Sample AMQSPUT0 start
target queue is HADRQ1
This is test 1
This is test 2
This is test 3
end

Sample AMQSPUT0 end
-bash-4.1$
```

Figure 8-13 Put test messages

- d. Shut off (or terminate) the primary QManager, with the following options:

- Search for all processes that are related to HADRQMGr:


```
ps -ef | grep HADRQMGr
```
- In the output, process *amqzxma0* has a parent PID of '1', whereas it is the parent for most of the other processes. Note the process ID, and terminate that process ID, using the following command (see Figure 8-14):

```
kill -9 <process-id>
```

```
-bash-4.1$ ps -Aef | grep HADRQMGr
mqm      14377      1      0 209351 16648      0 Oct04 ?        00:00:40 /opt/mqm/bin/amqzxma0 -m HADRQMGr -x
mqm      14382 14377      0 88384  4524      0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzfuma -m HADRQMGr
mqm      14388 14377      0 183383 14348      0 Oct04 ?        00:00:14 /opt/mqm/bin/amqzmu0 -m HADRQMGr
mqm      14393 14377      0 155492 3864      0 Oct04 ?        00:00:37 /opt/mqm/bin/amqzmu0 -m HADRQMGr
mqm      14394 14377      0 204140 5672      0 Oct04 ?        00:00:16 /opt/mqm/bin/amqzmu0 -m HADRQMGr
mqm      14397 14377      0 88476   6320      0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzrmfa -m HADRQMGr -t2332800 -s2592000
-p2592000 -g5184000 -c3600
mqm      14398 14377      0 77238   2800      0 Oct04 ?        00:00:01 /opt/mqm/bin/amqzdmaa -m HADRQMGr
mqm      14415 14377      0 70615   2828      0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzmg0 -m HADRQMGr
mqm      14425 14394      0 121706 5500      0 Oct04 ?        00:00:11 /opt/mqm/bin/amqzfpub -mHADRQMGr
mqm      14441 14377      0 202768 6768      0 Oct04 ?        00:01:29 /opt/mqm/bin/amqzlaa0 -mHADRQMGr -fip0
mqm      14442 14415      0 108317 2996      0 Oct04 ?        00:00:08 /opt/mqm/bin/runmqchi -m HADRQMGr -q SYSTEM.CHANNEL.IN
ITQ -r
mqm      14444 14415      0 74625  2812      0 Oct04 ?        00:00:08 /opt/mqm/bin/amqpcsea HADRQMGr
mqm      14449 14425      0 187215 6020      0 Oct04 ?        00:00:11 /opt/mqm/bin/amqfcxba -m HADRQMGr
mqm      14483 14415      0 85312  2724      0 Oct04 ?        00:00:02 /opt/mqm/bin/runmqqlr -r -m HADRQMGr -t TCP -p 1414
mqm      22030 12382      0 25809   840      0 18:54 pts/1  00:00:00 grep HADRQMGr
-bash-4.1$ kill -9 14377
-bash-4.1$ dspmqr
QMANAME (HADRQMGr)                                STATUS (Ended unexpectedly)
-bash-4.1$
```

Figure 8-14 Terminate (kill) the primary QManager

Note: This terminates the primary QManager processes. The standby then automatically becomes the primary.

- e. On the WMQ_Standby VM SSH session, check the QManager again and verify that it now runs as a primary using the **dspmqr** command. This shows the HADRQMgr running as primary:

```
QMNAME(HADRQMgr) STATUS (Running)
```

- f. Verify that you can retrieve the messages that were placed on the primary QManager before it was terminated using these commands:

```
cd /opt/mqm/samp/bin
```

```
./amqsget HADRQ1 HADRQMgr (this reads any message in the Queue)
```

The output shown in Figure 8-15 is displayed.

```
-bash-4.1$ cd /opt/mqm/samp/bin/
-bash-4.1$ ./amqsget HADRQ1 HADRQMgr
Sample AMQSGET0 start
message <This is test 1>
message <This is test 2>
message <This is test 3>
message <end>
no more messages
Sample AMQSGET0 end
-bash-4.1$ █
```

Figure 8-15 Get test messages

Note: This verifies that after putting the message in the primary queue and terminating the primary queue, the standby QManager takes over as the primary, and the messages are available in the new primary QManager. Starting the QManager in the old primary VM makes it the standby QManager.

8.3 Scenario WMQ_2: WebSphere MQ primary and standby in different patterns deployed on two different racks in the same data center

This scenario consists of two WebSphere MQ instances, one as the primary and the other as the standby for a given QManager.

On rack 1 (PDC-1), perform the following steps:

1. Build a pattern named RB_HADR_WMQ_2_Primary.
 - a. Drag a WebSphere MQ image part from the Image section of the palette. This part will be the primary QManager.
 - b. Name the part WMQ_Primary.
 - c. Provide the parameters to create the default QManager. On the WMQ_Primary part, add the following script packages with the parameters described in 8.1.3, “Script packages and parameters used in the pattern” on page 226:
 - WMQ-DelQMgr
 - WMQ-HAQMGR-Primary
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts

d. The pattern looks like Figure 8-16.

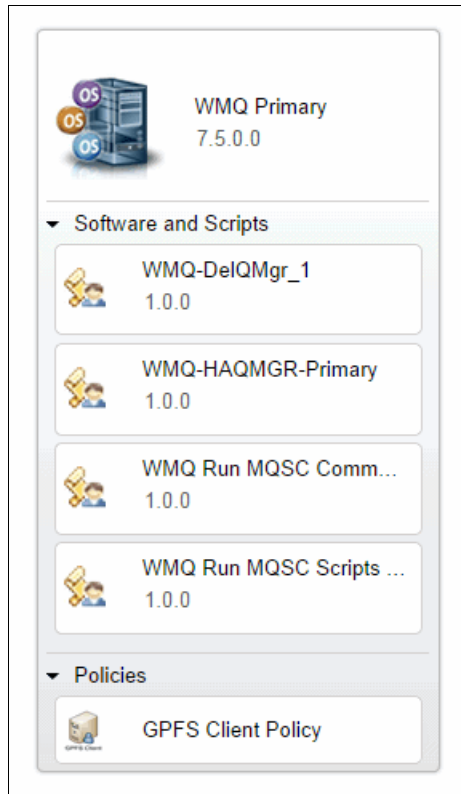


Figure 8-16 WebSphere MQ Primary pattern

The GPFS client connects through the GPFS shared service to the GPFS Primary-mirror setup described in 4.3, “Configuring an Active/Active (Mirrored) GPFS deployment” on page 53. The GPFS client policy parameters on the WebSphere MQ parts on both patterns look like Figure 8-17.

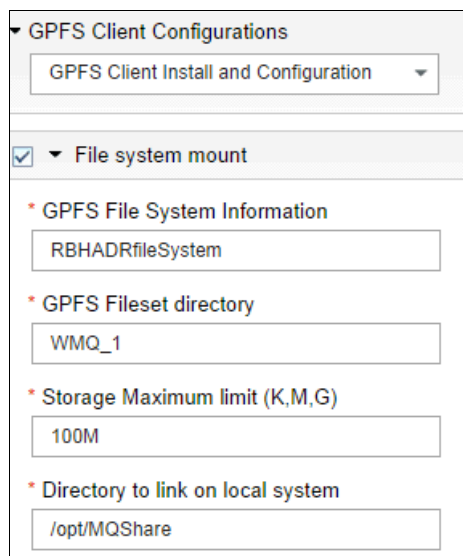
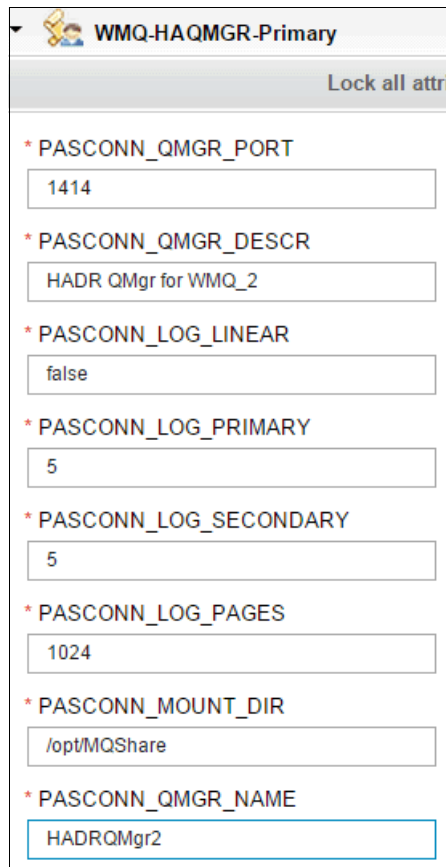


Figure 8-17 GPFS client policy configuration

2. Deploy the pattern using these steps:
 - a. From the virtual system pattern workload console, click the **Deploy** option for the virtual system RB HADR WMQ_1_Primary.
 - b. Provide the passwords for the WMQ_Primary part.
3. After the pattern is deployed, run the following scripts:
 - a. On *WMQ_Primary*, run WMQ-HAQMGR-Primary script with the parameters, shown in Figure 8-18.



| WMQ-HAQMGR-Primary | |
|-------------------------|---------------------|
| Lock all attri | |
| * PASCONN_QMGR_PORT | 1414 |
| * PASCONN_QMGR_DESCR | HADR QMgr for WMQ_2 |
| * PASCONN_LOG_LINEAR | false |
| * PASCONN_LOG_PRIMARY | 5 |
| * PASCONN_LOG_SECONDARY | 5 |
| * PASCONN_LOG_PAGES | 1024 |
| * PASCONN_MOUNT_DIR | /opt/MQShare |
| * PASCONN_QMGR_NAME | HADRQMGr2 |

Figure 8-18 HADRQMGr-Primary script

- b. On *WMQ_Primary*, run the *WebSphere MQ Run MQSC Command* script to create a local persistent queue with the parameters, shown in Figure 8-19. The MQSC command is **Define QLOCAL(HADRQ2) DEFPSIST(YES)**.

Fill in the required values for this script package

Required parameters value for executing script package

* QMGR_NAME: HADRQMgr2

* MQSC_COMMAND: Define QLOCAL(HADRQ2) DEFPSIST(Y)

Product administrator user name and password

User name: root

Password:

OK Cancel

Figure 8-19 *RunMQSCCommand* script

On **rack 2 (PDC-2)**, build, deploy, and get the environment for the standby WebSphere MQ VM by using these steps:

1. Build the pattern named **RB HADR WMQ_2 Standby** as shown below:
 - a. Drag a WebSphere MQ image part from the Image section of the palette to be the standby QManager, and name the part **WMQ_Standby**.
 - b. Provide the parameters to create the default QManager.
 - c. On the WebSphere MQ Standby part, add the following script packages with the parameters described in 8.1.3, “Script packages and parameters used in the pattern” on page 226:
 - WMQ-DelQMgr
 - WMQ-HAQMGR-Standby
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts

The pattern looks like Figure 8-20.

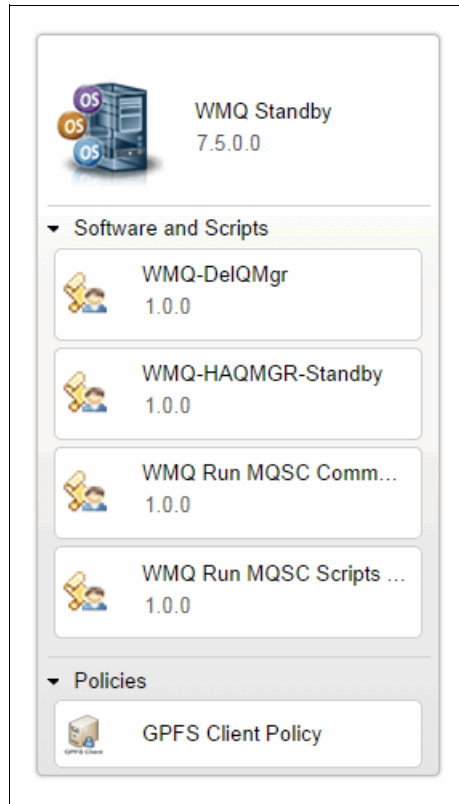


Figure 8-20 WebSphere MQ Standby pattern

2. The GPFS client policy parameters are the same as the primary WebSphere MQ pattern. See Figure 8-17 on page 236 for a listing of the parameters.
3. Deploy the pattern by using these steps:
 - a. From the virtual system pattern workload console, click the **Deploy** option for the virtual system RB HADR WMQ_2_Standby.
 - b. Provide the passwords for the WMQ_Standby part.
4. After it is deployed, run the WMQ-HAQMGR-Standby script on WMQ_Standby with the parameters shown in Figure 8-21.

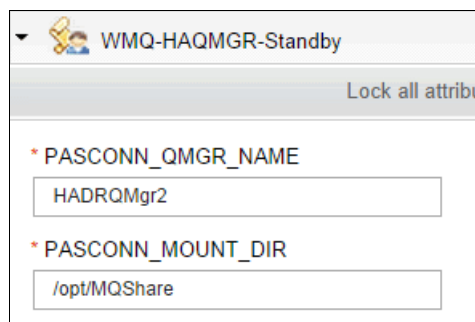


Figure 8-21 Parameters for WMQ-HAQMGR-Standby

The patterns are now deployed on the two racks (PDC-1 and PDC-2). The next steps are to verify the environment:

1. Send and receive messages.
 - a. Using PuTTY, open two SSH sessions, one to the WMQ_Primary VM and another to the WMQ_Standby VM. Use the IP addresses from the racks as shown in Figure 8-22.

| Name | Public IP | VM Status | CPU |
|--------------------------------|------------|-----------|-----|
| WMQ_Primary. 11412454482908 | 9.3.171.37 | Running | |

Figure 8-22 Primary virtual system instance

2. Check the QManagers running as primary and standby:
 - a. On WMQ_Primary VM SSH session, check the QManager by running the following commands:


```
su - mqm
dspmq (this shows the HADRQMgr2 running as primary)
QMNAME(HADRQMgr2) STATUS (Running)
```
 - b. On WMQ_Standby VM SSH session, check the QManager:


```
su - mqm
dspmq (this shows the HADRQMgr2 running as standby)
QMNAME (HADRQMgr2) STATUS (Running as standby)
```
3. On WMQ_Primary VM SSH session, put the test messages. This is similar to the example shown in Figure 8-13 on page 234, except the queue and QManager names in the command, as shown below:


```
cd /opt/mqm/samp/bin
./amqsput HADRQ2 HADRQMgr2 (this allows you to put the messages to the Queue)
This is test 1
This is test 2
This is test 3
End
<enter> (This ends the sample)
```
4. Shut off (or terminate) the primary QManager, with the following options:
 - a. Search for all processes that are related to HADRQMgr2:


```
ps -ef | grep HADRQMgr2
```
 - b. In the output, process amqzma0 has a parent PID of '1', whereas it is the parent for most of the other processes. Note the process ID, and terminate that process ID, similar to the figure shown in Figure 8-14 on page 234.


```
kill -9 <process-id>
```

Note: This terminates the primary QManager. The standby QManager then automatically becomes the primary.

5. On the WMQ_Standby VM SSH session, check the QManager again and verify that it now runs as a primary using the `dspm` command:

```
QMNAME(HADRQMgr2) STATUS (Running)
```

6. Verify that you can retrieve the messages placed on the primary QManager before it was terminated:

```
cd /opt/mqm/samp/bin
```

```
./amqsget HADRQ2 HADRQMgr2 (this reads any message in the Queue)
```

7. The output is similar to the output from scenario WMQ_1, as shown in Figure 8-15 on page 235.

Note: This verifies that after putting the message in the primary queue and terminating the primary queue, the standby QManager takes over as the primary and the messages are available in the new primary QManager. Starting the QManager in the old primary VM makes it the standby QManager.

8.4 Scenario WMQ_3: WebSphere MQ primary and standby in the different patterns deployed on two different racks across different data centers

This scenario consists of a minimum of two WebSphere MQ instances, one being the primary and the other being the standby for a specific QManager. The WebSphere MQ primary is deployed on the rack (PDC-1) in the primary data center, whereas the WebSphere MQ standby is deployed on the disaster recovery (SDC-1) rack at the second data center.

This setup uses a GPFS Passive configuration, where the GPFS Primary server is deployed on the primary rack (PDC-1) and the GPFS Passive server is deployed on the disaster recovery (DR) rack (SDC-1). Both servers use block storage. A replication pair is created between the two block storage volumes of the GPFS server, as shown in Figure 8-23.

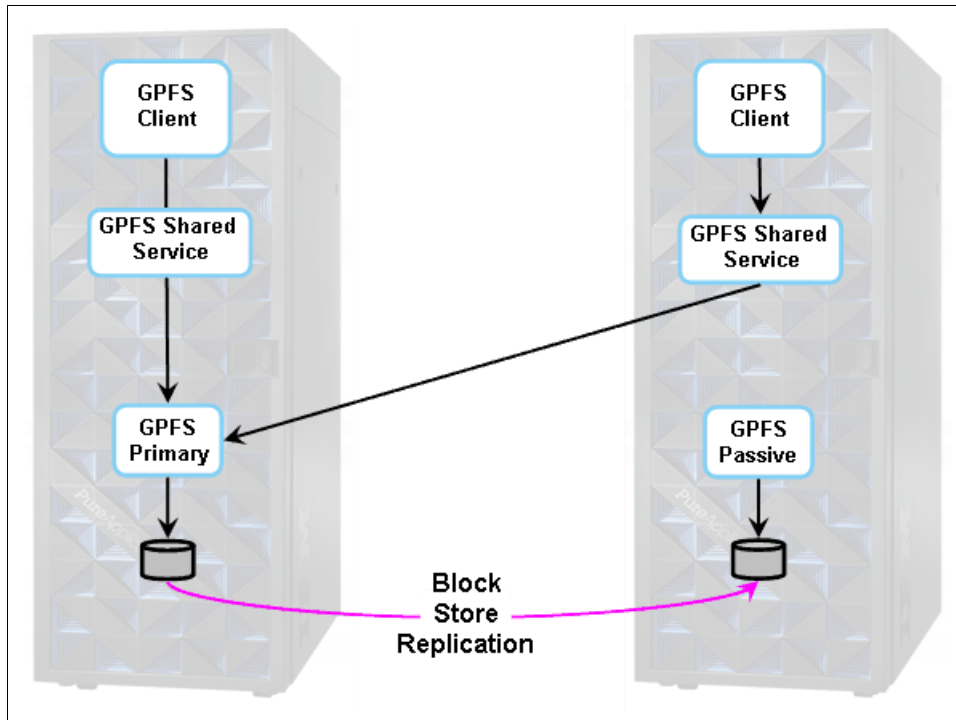


Figure 8-23 GPFS Passive before takeover

During the Passive takeover, complete the following manual steps are performed as shown here:

1. On GPFS Primary server, click **Prepare Primary for Takeover**.
2. Perform manual **Failover** operation on the Passive Block Storage volume. This action deletes the replication between the two block storage volumes.
3. On DR rack, for the GPFS Passive server, click **Passive Takeover**. This action makes the GPFS Passive server the primary server.
4. On DR rack, redeploy GPFS Shared Service to point to the new GPFS Primary.

After the Passive takeover, the topology looks like Figure 8-24.

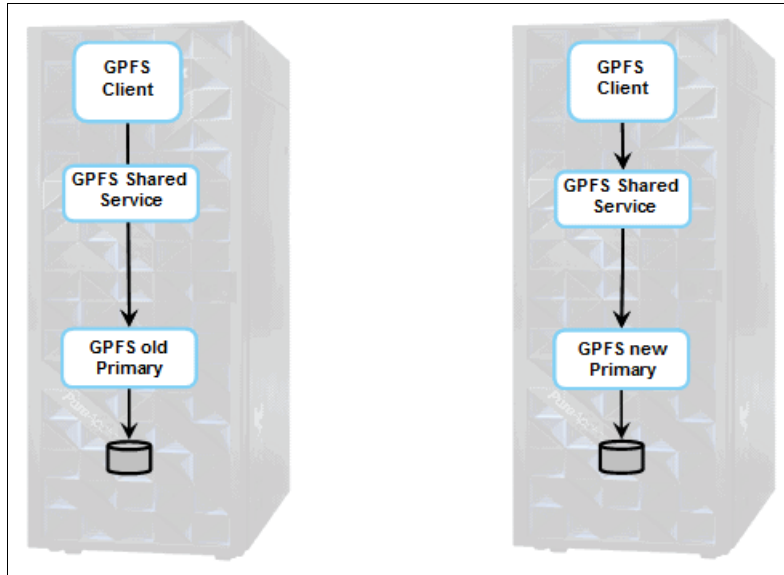


Figure 8-24 GPFS Passive after takeover

8.4.1 Steps for creating a WebSphere MQ active/passive scenario

On the primary rack (PDC-1) build, deploy, and set up the environment:

1. Build the pattern, named RB_HADR_WMQ_3_Primary, in the pattern builder editor.
 - a. Drag an WebSphere MQ image part from the Image section of the palette to be the standby QManager.
 - b. Name the part WMQ_Standby.
 - c. Provide the parameters to create the default QManager by adding the following script packages with the parameters described in “Script packages and parameters used in the pattern” on page 226.
 - WMQ-DelQMgr
 - WMQ-HAQMGR-Standby
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts

The GPFS client connects through the GPFS shared service to the GPFS Primary-passive setup described in 4.4, “Configuring an Active/Passive GPFS deployment” on page 61. The GPFS client policy parameters on the WebSphere MQ parts on both the patterns look like those in Figure 8-25.

The screenshot shows a configuration window titled "GPFS Client Configurations". At the top, there is a dropdown menu set to "GPFS Client Install and Configuration". Below this, a checkbox labeled "File system mount" is checked. Underneath, there are four fields, each preceded by an asterisk (*):

- * GPFS File System Information: RBHADRfileSystem
- * GPFS Fileset directory: WMQ_3
- * Storage Maximum limit (K,M,G): 100M
- * Directory to link on local system: /opt/MQShare

Figure 8-25 GPFS client policy

2. Deploy the pattern:
 - a. From the virtual system pattern workload console, click the **Deploy** option for the virtual system RB HADR WMQ_3 Primary.
 - b. Provide the passwords for the WMQ_Primary part.

3. After it is deployed, run the following scripts on WMQ_Primary:
- a. The WMQ-HAQMGR-Primary script with the parameters shown in Figure 8-26.

Fill in the required values for this script package

Required parameters value for executing script package

| | |
|-------------------------|---|
| * PASCONN_QMGR_PORT | <input type="text" value="1414"/> |
| * PASCONN_QMGR_DESCR | <input type="text" value="HADR Q Manager"/> |
| * PASCONN_LOG_LINEAR | <input type="text" value="false"/> |
| * PASCONN_LOG_PRIMARY | <input type="text" value="5"/> |
| * PASCONN_LOG_SECONDARY | <input type="text" value="5"/> |
| * PASCONN_LOG_PAGES | <input type="text" value="1024"/> |
| * PASCONN_MOUNT_DIR | <input type="text" value="/opt/MQShare"/> |
| * PASCONN_QMGR_NAME | <input type="text" value="HADRQMgr3"/> |

Product administrator user name and password

User name:

Password:

Figure 8-26 HADRQMgr-Primary script

- b. The WebSphere MQ Run MQSC Command script to create a local persistent queue, with the parameters shown in Figure 8-27 (the MQSC command is **Define QLOCAL(HADRQ3) DEFPSIST(YES)**).

Fill in the required values for this script package

Required parameters value for executing script package

* QMGR_NAME: HADRQMGr3

* MQSC_COMMAND: DEFINE QLOCAL(HADRQ3) DEFPSIST

Product administrator user name and password

User name: root

Password:

OK Cancel

Figure 8-27 WebSphere MQ Run MQSC Command script

On **rack 2 (PDC-2)**, build, deploy, and get the environment for the standby WebSphere MQ standby VM.

4. Build the pattern named RB HADR WMQ_3 Passive Standby in the pattern builder editor.
 - a. Drag a WebSphere MQ image part from the Image section of the palette to be the standby QManager.
 - b. Name the part WMQ_Standby.
 - c. Provide the parameters to create the default QManager. Add the following script packages with the parameters described in “Script packages and parameters used in the pattern” on page 226:
 - WMQ-DelQMGr
 - WMQ-HAQMGR-Standby
 - WebSphere MQ Run MQSC Command
 - WebSphere MQ Run MQSC Scripts

The pattern looks like Figure 8-28.

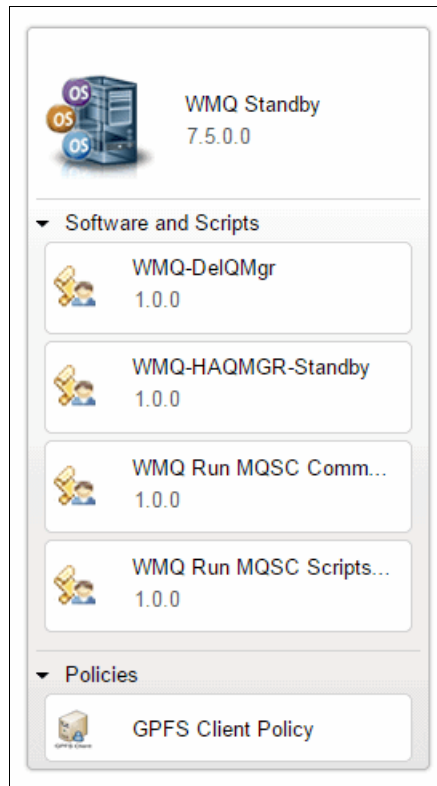


Figure 8-28 WMQ_Standby pattern

The GPFS client policy parameters are the same as the primary WebSphere MQ pattern, as shown in Figure 8-25 on page 244.

5. Deploy the pattern.
 - a. From the virtual system pattern workload console, click the **Deploy** option for the virtual system pattern RB HADR WMQ_3 Passive Standby.
 - b. Provide the passwords for the WMQ_Standby part.

- After it is deployed, run the WMQ-HAQMGR-Standby script on WMQ_Standby with the parameters that are shown in Figure 8-29.

Fill in the required values for this script package

Required parameters value for executing script package

* PASCONN_QMGR_NAME

* PASCONN_MOUNT_DIR

Product administrator user name and password

User name:

Password:

Figure 8-29 HADRMgr_Standby_Script

Now that the patterns are deployed on the two racks, the high-level steps are as follows:

- From primary WebSphere MQ, send the message to the queue.
- Terminate (kill) the WebSphere MQ primary process.
- Perform GPFS passive failover.
- From WMQ_Standby, receive the messages that were sent from WMQ_Primary.

These are the detailed steps based on the high-level steps:

- From primary WebSphere MQ, send the message to the queue:
 - Using PuTTY, open SSH sessions to the WebSphere MQ Primary VMs each. Use the IP addresses from the Primary rack, as shown in the virtual instance console for this deployment, as shown in Figure 8-30.

RB HADR WMQ_2 Primary Refresh

Virtual machine perspective (1 in total)

| | Name | Public IP | VM Status | CPI |
|---|--------------------------------|------------|-----------|-----|
| + | WMQ_Primary. 11412454482908 | 9.3.171.37 | ▶ Running | ■ |

Figure 8-30 Primary rack IP addresses

- Check the QManagers on the primary:
 - On WMQ_Primary VM SSH session, run the following commands to check the QManager:


```
su - mqm (change the user to mqm)
```

```
dspmq (this command shows HADRQMgr3 running as primary, as shown below)
```

QMNAME(HADRQMgr3) STATUS (Running)

- b. On WMQ_Primary VM SSH session, put the test messages as follows:

```
cd /opt/mqm/samp/bin
```

- c. Use the `./amqsput HADRQ3 HADRQMgr3` command to put the messages to the Queue:

```
This is test 1
```

```
This is test 2
```

```
This is test 3
```

```
End
```

```
<enter> (this ends the sample)
```

This procedure is similar to Figure 8-31.

```
login as: root
root@9.3.171.36's password:
Last login: Sun Oct 19 17:05:53 2014 from sig-9-65-4-221.mts.ibm.com
IBM WebSphere MQ Hypervisor Edition for Red Hat Enterprise Linux
[root@ipas-lpar-9-3-171-36 ~]# su - mqm
-bash-4.1$ cd /opt/mqm/samp/bin/
-bash-4.1$ ./amqsput HADRQ1 HADRQMgr
Sample AMQSPUTO start
target queue is HADRQ1
This is test 1
This is test 2
This is test 3
end

Sample AMQSPUTO end
-bash-4.1$
```

Figure 8-31 Put the messages

3. Terminate (kill) the WebSphere MQ primary process:

- a. Search for all processes related to HADRQMgr3 by using the `ps -ef | grep HADRQMgr3` command. In the output, process amqzma0 has a parent PID of '1', whereas it is the parent for most of the other processes.

- b. Note the process ID.

- c. Terminate the process ID as shown in Figure 8-32:

```
kill -9 <process-id>
```

This command terminates the primary QManager.

```
-bash-4.1$ ps -Aef | grep HADRQMgr
mqm      14377      1      0 209351 16648  0 Oct04 ?        00:00:40 /opt/mqm/bin/amqzma0 -m HADRQMgr -x
mqm      14382 14377  0 88384  4524  0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzfuma -m HADRQMgr
mqm      14388 14377  0 183383 14348  0 Oct04 ?        00:00:14 /opt/mqm/bin/amqzmuc0 -m HADRQMgr
mqm      14393 14377  0 155492 3864  0 Oct04 ?        00:00:37 /opt/mqm/bin/amqzmur0 -m HADRQMgr
mqm      14394 14377  0 204140 5672  0 Oct04 ?        00:00:16 /opt/mqm/bin/amqzmf0 -m HADRQMgr
mqm      14397 14377  0 88476  6320  0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzrmfa -m HADRQMgr -t2332800 -s2592000
-p2592000 -g5184000 -c3600
mqm      14398 14377  0 77238  2800  0 Oct04 ?        00:00:01 /opt/mqm/bin/amqzdmaa -m HADRQMgr
mqm      14415 14377  0 70615  2828  0 Oct04 ?        00:00:05 /opt/mqm/bin/amqzmgr0 -m HADRQMgr
mqm      14425 14394  0 121706 5500  0 Oct04 ?        00:00:11 /opt/mqm/bin/amqzfpub -mHADRQMgr
mqm      14441 14377  0 202768 6768  0 Oct04 ?        00:01:29 /opt/mqm/bin/amqzlaa0 -mHADRQMgr -fip0
mqm      14442 14415  0 108317 2996  0 Oct04 ?        00:00:08 /opt/mqm/bin/runmqchi -m HADRQMgr -q SYSTEM.CHANNEL.IN
ITQ --r
mqm      14444 14415  0 74625  2812  0 Oct04 ?        00:00:08 /opt/mqm/bin/amqpcsea HADRQMgr
mqm      14449 14425  0 187215 6020  0 Oct04 ?        00:00:11 /opt/mqm/bin/amqfcxba -m HADRQMgr
mqm      14483 14415  0 85312  2724  0 Oct04 ?        00:00:02 /opt/mqm/bin/runmqslsr -r -m HADRQMgr -t TCP -p 1414
mqm      22030 12392  0 25809   840  0 18:54 pts/1  00:00:00 grep HADRQMgr
-bash-4.1$ kill -9 14377
-bash-4.1$ dspmq
QMNAME(HADRQMgr)                                STATUS(Ended unexpectedly)
-bash-4.1$
```

Figure 8-32 Terminate the process

The next set of steps are to enable the WebSphere MQ standby on the SDC rack (DR rack):

1. On rack 1 (PDC-1) for the GPFS Primary Server:
 - a. From the Server instance console, select **GPFS Manager** → **Manage** → **Operations**.
 - b. Perform **Prepare Primary For Takeover** as shown in Figure 8-33. This would be done for a planned failover. In case of an unplanned failover, this might not be an option and you would start with the next step of passive takeover.

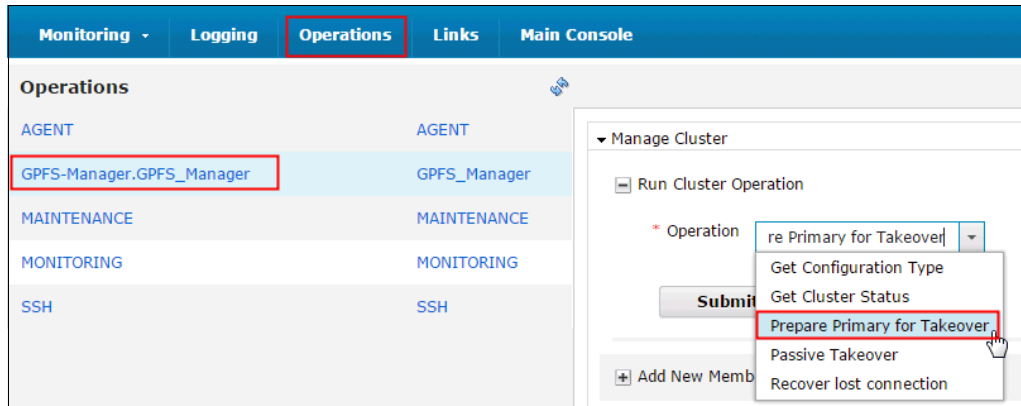


Figure 8-33 Preparing primary for takeover

2. On rack 2 (SDC-1) for the GPFS Passive Server:
 - a. From the Server instance console, select **GPFS Manager** → **Manage** → **Operations**.
 - b. Perform the **Passive Takeover** as shown in Figure 8-34.

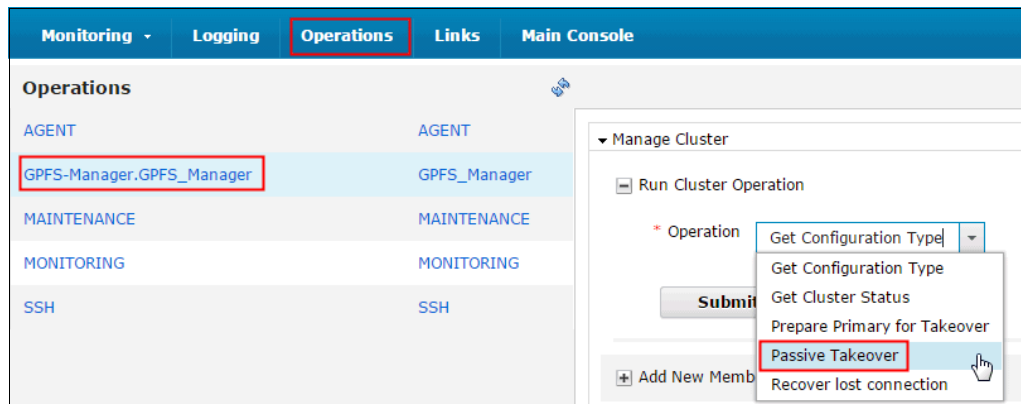


Figure 8-34 Passive takeover

3. On rack 2 (SDC-1), complete these steps:
 - a. Delete the GPFS Shared service that was pointing to the Primary rack GPFS Primary server.
4. On rack 2 (SDC-1), complete these steps:
 - a. Get the new GPFS Primary Server's (original Passive Server's) client key.
 - b. Use the client key to redeploy the GPFS shared service.
5. On rack 2 (SDC-1), complete these steps:
 - a. Go to the WebSphere MQ standby server instance.
 - b. For the WMQ_Standby GPFS Client, select **Manage** → **Operations**.

- c. Perform **Connect to Server** as shown in Figure 8-35.

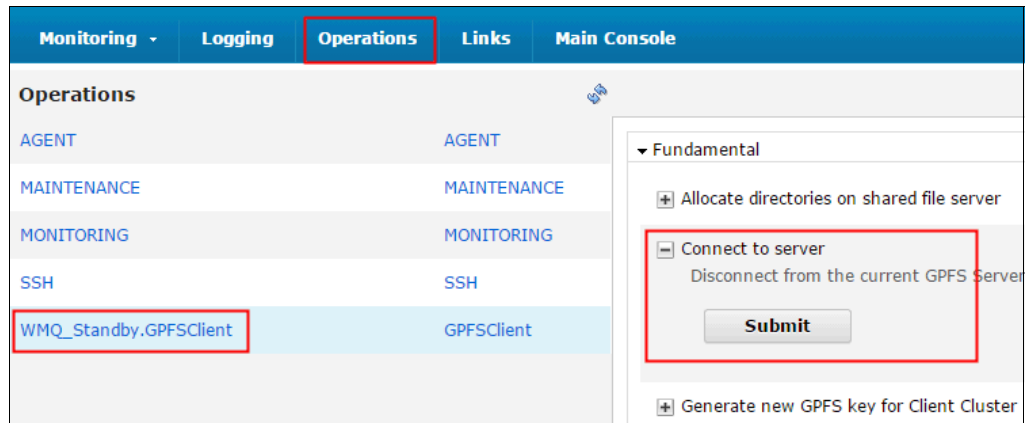


Figure 8-35 Connect to server

The GPFS Passive takes over and the clients on the DR rack are connected to the new GPFS Primary server with the QManager data from the old primary GPFS server.

6. On the WMQ_Standby VM SSH session, check the QManager again and verify that it now runs as a primary:

su - mqm (change the user to mqm)

dsqm (this command shows the HADRQMgr3 running as primary)

QMNAME(HADRQMgr3) STATUS (Running)

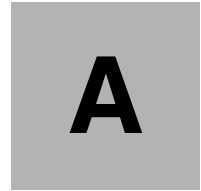
7. Verify that you can retrieve the messages that were placed on the primary QManager:

cd /opt/mqm/samp/bin

./amqsget HADRQ3 HADRQMgr3 (this command reads any message in the Queue)

The output that is displayed is similar to Figure 8-15 on page 235.

Note: This procedure verifies that after putting the message in the primary queue, terminating the primary queue, and completing the GPFS Passive takeover, the standby QManager takes over as the primary, and the messages are available in the new primary QManager.



Sample Application

This appendix introduces the sample program that was used to test the use case scenarios in this IBM Redbooks publication. The sample application (see Figure A-1) was developed to simulate a bank account transaction.



Figure A-1 Sample program handling bank account transaction

The application uses XA Data Source in a transaction that is viewed as a set of tasks run as a unit of work. XA Data Sources are Java Transaction API (JTA) compliant and allow the resource manager to support distributed transaction management. In this example, an amount is transferred from one bank account to another account. The two operations must succeed to complete the entire money transfer transaction. If the two operations are successfully completed, the transaction is committed. If not, the transaction is rolled back and neither operation takes place.

Because the sample application requires access to a relational database to fulfill its business objective, databases and tables for bank accounts must be created. Two databases for each table were created. A data source was created that allows the application to connect and use the database. After creating database tables through the DB2 command line processor and configuring the data source from the administrative console, the application was deployed and tested.

The following sections are covered in this appendix:

- ▶ Database Setup
- ▶ Application Coding Explanation
- ▶ Testing

Database Setup

Two databases are created so that the sample program can simulate a distributed transaction by connecting and accessing two different database resources.

The next sections cover the following requirements:

- ▶ Create CHECKING database and SAVINGS database
- ▶ List database directory
- ▶ Create tables
- ▶ Insert a single row into the SAVINGS table
- ▶ Terminate and configure CHECKING database and table

Create CHECKING database and SAVINGS database

Use SSH to connect to the Database server instance as `db2inst1`. Issue the `db2 create database` command, as shown in Example A-1.

Example A-1 Create database CHECKING and SAVINGS

```
-bash-4.1$ db2 create database CHECKING
DB20000I The CREATE DATABASE command completed successfully.
-bash-4.1$ db2 create database SAVINGS
DB20000I The CREATE DATABASE command completed successfully.
-bash-4.1$
```

List database directory

After creating the databases, you need to verify that the database has been created by listing the system database directory contents. Issue the `list db directory` command, as shown in Example A-2.

Example A-2 Verify that the databases are created

```
-bash-4.1$ db2 list db directory

System Database Directory

Number of entries in the directory = 2

Database 1 entry:

Database alias           = SAVINGS
Database name           = SAVINGS
Local database directory = /db2fs
Database release level  = f.00
Comment                 =
Directory entry type    = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =

Database 2 entry:

Database alias           = CHECKING
```

| | |
|-----------------------------------|------------|
| Database name | = CHECKING |
| Local database directory | = /db2fs |
| Database release level | = f.00 |
| Comment | = |
| Directory entry type | = Indirect |
| Catalog database partition number | = 0 |
| Alternate server hostname | = |
| Alternate server port number | = |

Create tables

A table is an unsorted set of records that consist of rows and columns. Each column has a defined data type, and each row represents an entry in the table. You need to create both a SAVINGS and a CHECKING table, and each table must contain the account number, name, and balance amount. The sample application updates the balance amount value according to the transaction type. The Account number is used as a primary key that uniquely identifies records in the table.

To create tables from each database, complete the following steps and refer to Example A-3:

1. Create a table called SAVINGS from the database called SAVINGS.
2. From the SSH window, enter **DB2** and then enter **connect to SAVINGS**.
3. Issue the **CREATE TABLE SAVINGS (ACCOUNT_NUMBER INT NOT NULL PRIMARY KEY, NAME VARCHAR(20) NOT NULL, AMOUNT DECIMAL(7,2))** command as shown in Example A-3.

Example A-3 Create table SAVINGS from database SAVINGS

```
-bash-4.1$ db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 10.1.4
```

You can issue database manager commands and SQL statements from the command prompt. For example:

```
db2 => connect to sample
db2 => bind sample.bnd
```

For general help, type: ?.

For command help, type: ? command, where command can be the first few keywords of a database manager command. For example:

```
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.
```

To exit db2 interactive mode, type QUIT at the command prompt. Outside interactive mode, all commands must be prefixed with 'db2'. To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

```
db2 => connect to savings
```

```
Database Connection Information
```

```
Database server      = DB2/LINUX8664 10.1.4
SQL authorization ID = DB2INST1
Local database alias = SAVINGS
```

```
db2 => CREATE TABLE SAVINGS (ACCOUNT_NUMBER INT NOT NULL PRIMARY KEY, NAME
VARCHAR(20) NOT NULL, AMOUNT DECIMAL(7,2))
DB20000I The SQL command completed successfully.
```

Note: As you continue through this section, you will see information in “Terminate and configure CHECKING database and table” on page 257 to address similar needs for the CHECKING database.

Insert a single row into the SAVINGS table

Because the sample program uses the specific ACCOUNT_NUMBER, you need to initialize the data by inserting a row (record). After inserting a row, verify the data in the row that you inserted by querying it. Example A-4 provides the database command to insert a single row into the SAVINGS table. The sample program uses a specific account number, 12345, for SAVINGS and 54321 for CHECKING. You need to initialize both accounts. You can initialize the balance amount value with any positive number.

Example A-4 Initialize a SAVINGS account with ACCOUNT_NUMBER “12345”.

```
db2 => INSERT INTO SAVINGS (ACCOUNT_NUMBER,NAME,AMOUNT) VALUES
(12345,'IPAS',20000.00)
DB20000I The SQL command completed successfully.
db2 => select * from SAVINGS
```

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|-------|----------|
| ----- | ----- | ----- |
| 12345 | IPAS | 20000.00 |

1 record(s) selected.

```
db2 =>
```

Terminate and configure CHECKING database and table

Use similar steps to create the same structures in the CHECKING database. Because there is a connection to the SAVINGS database, use the following steps to terminate the connection to the SAVINGS database and then connect to the CHECKING database as shown in Example A-5:

1. Issue the **terminate** command to delete the connection to the database, SAVINGS.
2. Connect to the database, CHECKING, to create a table called CHECKING.
3. Insert a row with the CHECKING account's ACCOUNT_NUMBER, which is 54321.

Example A-5 Terminate SAVINGS database connection and connect to CHECKING for data setup

```
db2 => quit
DB20000I The QUIT command completed successfully.
-bash-4.1$ db2 terminate
DB20000I The TERMINATE command completed successfully.
-bash-4.1$ db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 10.1.4
```

You can issue database manager commands and SQL statements from the command prompt. For example:

```
db2 => connect to sample
db2 => bind sample.bnd
```

For general help, type ?.

For command help, type ? command, where command can be the first few keywords of a database manager command. For example:

```
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG           for help on all of the CATALOG commands.
```

To exit db2 interactive mode, type QUIT at the command prompt. Outside interactive mode, all commands must be prefixed with 'db2'. To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

```
db2 => connect to checking
```

Database Connection Information

```
Database server      = DB2/LINUX8664 10.1.4
SQL authorization ID = DB2INST1
Local database alias = CHECKING
```

```
db2 => CREATE TABLE CHECKING (ACCOUNT_NUMBER INT NOT NULL PRIMARY KEY, NAME
VARCHAR(20) NOT NULL, AMOUNT DECIMAL(7,2))
```

```
DB20000I The SQL command completed successfully.
```

```
db2 => INSERT INTO CHECKING (ACCOUNT_NUMBER,NAME,AMOUNT) VALUES
(54321,'IPAS',8000.00)
```

```
DB20000I The SQL command completed successfully.
```

```
db2 => select * from checking
```

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|---------|
| 54321 | IPAS | 8000.00 |

```
1 record(s) selected.
```

```
db2 =>
```

Application Coding Explanation

The application uses the UserTransaction interface, which provides the application the ability to programmatically control transaction boundaries. UserTransaction can be obtained from the SessionContext through `java:comp/UserTransaction`, as shown in Example A-6.

Example A-6 Method to obtain the UserTransaction

```
static UserTransaction getUserTransaction(HttpServletRequest resp) throws
IOException{
    //printHTML(resp,"In getUserTransaction");
```

```

String msg;
UserTransaction utx = null;
try {
    Context initialContext = new InitialContext();
    utx = (UserTransaction) initialContext.lookup("java:comp/UserTransaction");

}
catch ( NamingException ex ) {
    msg="Cannot get UserTransaction: " + ex;
    printHTML(resp,msg);
}

return utx;
}

```

The **UserTransaction.begin()** method creates a new global transaction and associates it with the current calling thread. The **UserTransaction.commit()** method completes the transaction that is associated with the current thread. When this method completes, the thread is no longer associated with a transaction. The **UserTransaction.rollback()** method rolls back the transaction that is associated with the current thread. When this method completes, the thread is no longer associated with a transaction. Example A-7 shows these methods. In this example, **utx** is used instead of the long form of the word, **UserTransaction**.

Example A-7 The UserTransaction methods invocation

```

UserTransaction utx=null;
utx = getUserTransaction(response);
:
try {
    utx.begin();
    :
    utx.commit();

}catch (SQLException s){
:
    utx.rollback();
}
}

```

First phase (withdraw)

When a user transfers a certain amount from one account to another, the application retrieves the existing balance of the account to withdraw.

If there not sufficient funds to withdraw, the program sends an error message and ends execution.

If there are sufficient funds to withdraw, the first transaction deducts the balance by the amount specified, updates the balance, and moves to the second phase.

If there is any error situation, the transaction is rolled back.

Second phase (deposit)

After the first phase succeeds, the application waits for one minute before running the second phase of the transaction (deposit). The application retrieves the current balance of the account, adds the amount, and updates the balance. After those two phases are successful, the transaction is committed. If the second phase fails, the entire transaction is rolled back.

Testing

There are two bank accounts, Savings and Checking. The Savings account uses the SAVINGS database. The Checking account uses the CHECKING database. Figure A-2 shows the empty balances.

| Savings Account Balance | | | Checking Account Balance | | |
|-------------------------|------|--------|--------------------------|------|--------|
| ACCOUNT_NUMBER | NAME | AMOUNT | ACCOUNT_NUMBER | NAME | AMOUNT |
| 12345 | IPAS | 0.00 | 54321 | IPAS | 0.00 |

Figure A-2 Initialize both accounts as zero

Sample Testing

You are now ready to test the two-phase commit transaction. This test verifies that an application properly allows a user to transfer balance from one bank account to another. When the user transfers a certain amount of balance, the first account must be deducted that amount and then the second account must be increased that amount.

Use the following steps to test the application:

1. Deposit \$50,000.00 to the SAVINGS account. See Figure A-3.

| Choose Transaction: | DEPOSIT to SAVINGS | Amount: | 50000 |
|---|--------------------|----------|-------|
| <input type="button" value="Submit Query"/> | | | |
| Savings Account Balance | | | |
| ACCOUNT_NUMBER | NAME | AMOUNT | |
| 12345 | IPAS | 50000.00 | |

Figure A-3 Deposit to SAVINGS account

- Transfer \$20,000.00 from the SAVINGS account to the CHECKING account. See Figure A-4.

Choose Transaction: TRANSFER from SAVINGS to CHECKING Amount: 20000

Submit Query

Savings account balance

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|----------|
| 12345 | IPAS | 30000.00 |

Checking account balance

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|----------|
| 54321 | IPAS | 20000.00 |

Figure A-4 Transfer from SAVINGS account to CHECKING account

- Withdraw \$30,000 from the SAVINGS account. See Figure A-5.

Choose Transaction: WITHDRAW from SAVINGS Amount: 30000

Submit Query

Savings Account Balance

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|--------|
| 12345 | IPAS | 0.00 |

Figure A-5 Withdraw from the SAVINGS account

- Transfer \$10,000 from the CHECKING account to the SAVINGS account. See Figure A-6.

Choose Transaction: TRANSFER from CHECKING to SAVINGS Amount: 10000

Submit Query

Checking account balance

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|----------|
| 54321 | IPAS | 10000.00 |

Savings account balance

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|----------|
| 12345 | IPAS | 10000.00 |

Figure A-6 Transfer from CHECKING to SAVINGS

- To test the rollback response to a failure of the first transaction, drop the CHECKING database CHECKING table from DB2. Issue the **DB2 CLP** command as shown in Example A-8.

Example A-8 Drop the CHECKING table

```
db2 => connect to checking

      Database Connection Information

Database server      = DB2/LINUX8664 10.1.4
SQL authorization ID = DB2INST1
Local database alias = CHECKING

db2 => drop table checking
DB20000I The SQL command completed successfully.
db2 =>
```

- Transfer \$1,000 from the SAVINGS account to the CHECKING account. See Figure A-7.

Figure A-7 Roll-back test

- This is a two-phase transaction and the second phase failed. The first phase that deducted \$1,000 from the savings must be rolled back. To check the balance of the SAVINGS account, deposit \$0.00 as shown in Figure A-8. The balance will be \$10,000.00 if the rollback process was successful.

| ACCOUNT_NUMBER | NAME | AMOUNT |
|----------------|------|----------|
| 12345 | IPAS | 10000.00 |

Figure A-8 Roll-back test

8. Re-create the table for CHECKING, if you want to continue other testing. Because the application specifically handles CHECKING account number 54321, you need to initialize the record by inserting a row with ACCOUNT_NUMBER = 54321, as shown in Example A-9.

Example A-9 Re-create table CHECKING and initialize the record

```
db2 => CREATE TABLE CHECKING (ACCOUNT_NUMBER INT NOT NULL PRIMARY KEY, NAME
VARCHAR(20) NOT NULL, AMOUNT DECIMAL(7,2))
DB20000I The SQL command completed successfully.
db2 => INSERT INTO CHECKING (ACCOUNT_NUMBER,NAME,AMOUNT) VALUES
(54321,'IPAS',0.00)
DB20000I The SQL command completed successfully.
db2 =>
```



B

Common WebSphere Application Server configuration tasks

This appendix provides detailed instructions to perform common WebSphere Application Server configuration operations needed in this book. These instructions are provided as a convenience to the reader and should only be used in the context of this publication.

The following tasks are covered in this appendix:

- ▶ Build the WebSphere Application Server cluster pattern
- ▶ Deploy WebSphere Application Server cluster pattern: Single rack
- ▶ Deploy WebSphere Application Server cluster pattern: Multiple rack
- ▶ Create WebSphere Application Server cluster
- ▶ Configure database connectivity for BankTransaction application
- ▶ Install BankTransaction application
- ▶ Validate BankTransaction application

Build the WebSphere Application Server cluster pattern

The steps necessary to build the WebSphere Application Server cluster pattern are shown in this section. The pattern provides a Deployment Manager, Custom Nodes, and IBM HTTP Server.

Note: The pattern name for this example is RB_HADR_WAS_1B. Be sure to implement an appropriate pattern name.

1. From the Workload Console, select **Patterns** → **Virtual System Patterns**.
2. Click **Create New** to start building a pattern.
 - a. Provide a name for the pattern. The example shown here is RB_HADR_WAS_1B.
 - b. The wizard shows a list of pre-built templates that are included with the WebSphere Application Server component.
 - c. Select the template called **WebSphere Application Server Cluster**. This template provides a Deployment Manager, Custom Nodes, and IBM HTTP Server.
 - d. Click **Start Building** as shown in Figure B-1.

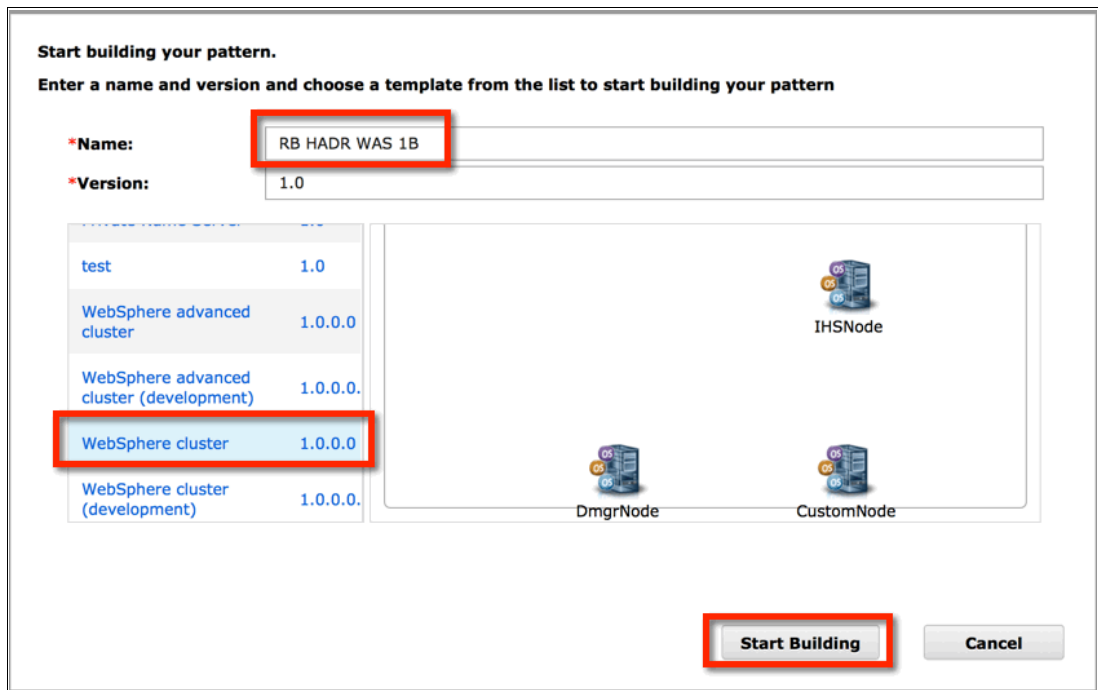


Figure B-1 Build your pattern

3. After a brief time, a notification indicates that the new pattern was created.

4. Select **Patterns** → **Virtual System Patterns** and filter for example, RB HADR. You should see your pattern listed as shown in Figure B-2.

| Name | Version | Status | Created by | Updated by | Created on | Updated on | Actions |
|----------------|---------|--------|------------|------------|--------------------------|--------------------------|---------|
| RB HADR WAS 1B | 1.0 | Draft | vgadepal@ | vgadepal@ | Oct 23, 2014, 5:10:31 PM | Oct 23, 2014, 5:10:40 PM | [Icons] |
| RB HADR WMQ_1 | 1.0 | Draft | rgandhi@u: | rgandhi@u: | Sep 30, 2014, 6:48:56 PM | Oct 4, 2014, 11:37:32 AM | [Icons] |

Figure B-2 Verify pattern creation

5. Now that the pattern is created, edit and modify it, complete these steps:
 - a. Click the **Edit** icon to open the pattern in the pattern editor.
 - b. The pattern builder opens in a new window. It should look similar to Figure B-3.

Figure B-3 Pattern builder

6. Review and lock the pattern level parameters:
 - a. You can specify a password for root and virtuser either in the pattern editor or at deployment time.
 - b. For the WebSphere administrative user name, you can use virtuser or use whatever name you want. This case uses virtuser.

- c. The password for the WebSphere administrative user can also be specified in the pattern editor or left for the deployer to specify at deployment time as shown in Figure B-4.

The screenshot shows a web form titled "Pattern-level Parameters". At the top left, the title is enclosed in a red box. Below the title is a button labeled "+ Add new parameter". The form contains four main sections, each with a red asterisk indicating a required field:

- Password (root)**: Two input fields, both containing "*****".
- Password (virtuser)**: Two input fields, both containing "*****".
- WebSphere administrative user name**: A single input field containing the text "virtuser". This field and its label are enclosed in a red box.
- WebSphere administrative password**: Two input fields, both containing "*****".

Figure B-4 Pattern parameters

- d. Leave the default values for the component parts in the pattern.
7. Save the pattern by clicking **Save**.
8. Add scripts to the pattern to customize it.
 - a. Expand the Scripts section the left pane.
 - b. Find the script **Disable IPTables** by typing IPTables in the filter field.
 - c. Drag the script **Disable IPTables** onto **DmgrNode**, **CustomNode**, and **IHSNode**.

Note: The `Disable IPTables` script shuts down the firewall on each of the VMs. This is not recommended for customers, but has been done here to simplify the scenarios in this publication. A better option is to have the script open only the ports that are necessary. This script is run at deployment time.

9. The pattern should resemble Figure B-5. Click **Save**.

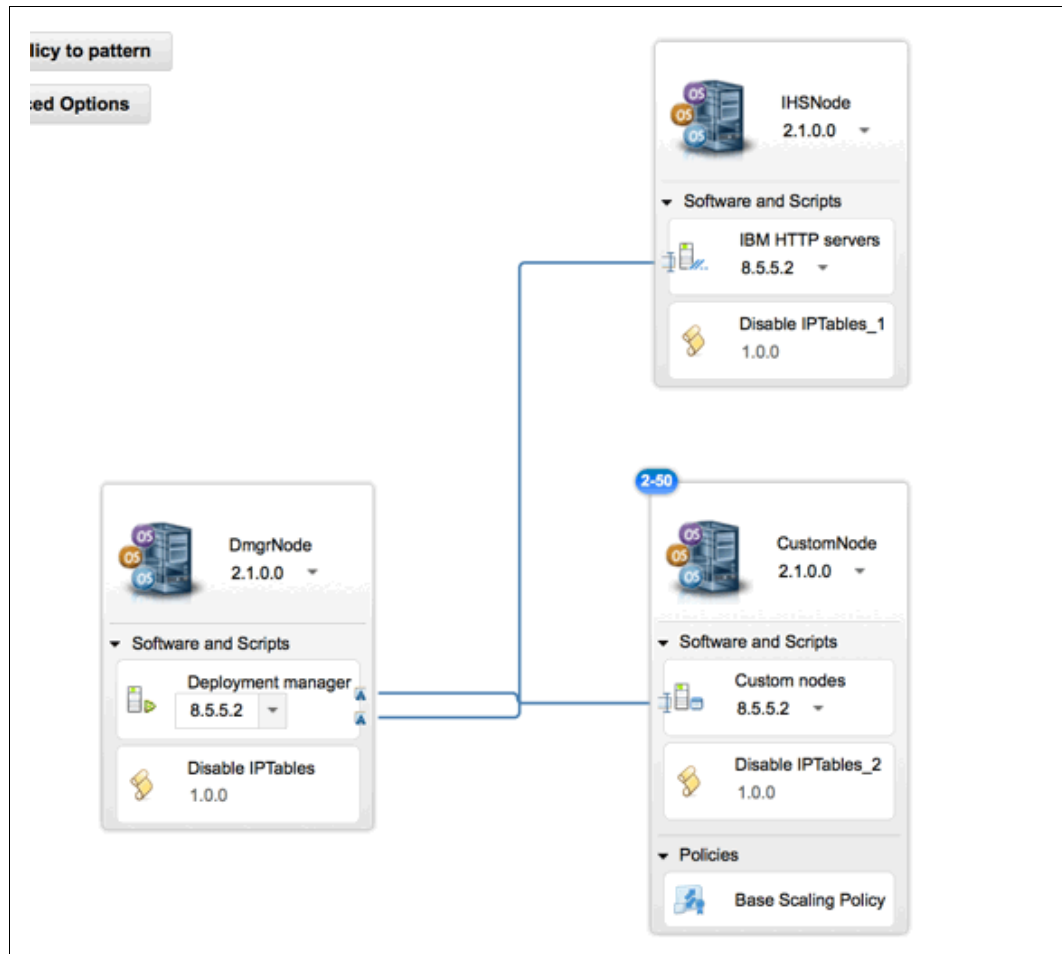


Figure B-5 Add scripts to pattern parts

Deploy WebSphere Application Server cluster pattern: Single rack

These instructions provide details about deploying the pattern created in “Build the WebSphere Application Server cluster pattern” on page 266. Before continuing, verify the name of the pattern you want to deploy. This deployment is for a **single rack only**.

1. Click **Deploy** to start the deployment of the virtual system pattern.
2. Provide the password for the VMs. This pattern deployment creates four VMs (Deployment Manager, two Custom Nodes, and one IBM HTTP Server).

3. Check the status of the deployment by looking in the **Instances** → **Virtual System Instances** page as shown in Figure B-6.

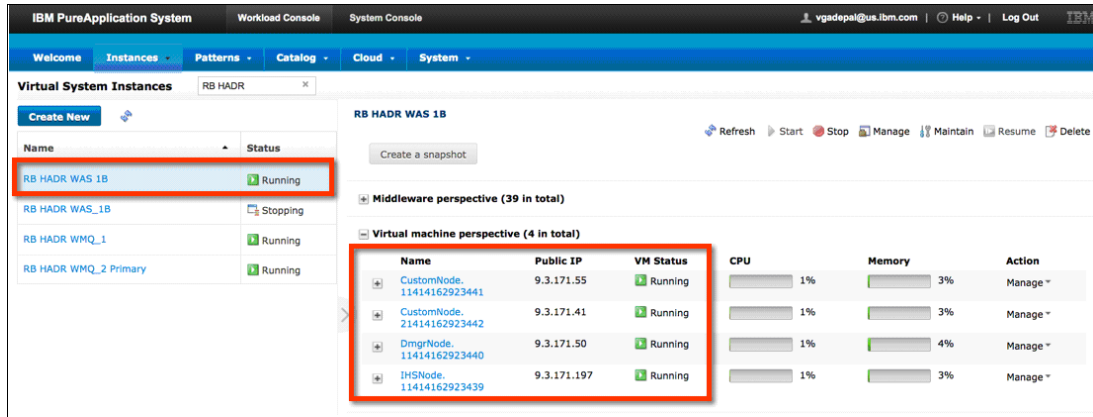


Figure B-6 Pattern status

Deploy WebSphere Application Server cluster pattern: Multiple rack

These instructions provide details about deploying the pattern that was created in “Build the WebSphere Application Server cluster pattern” on page 266. Before continuing, verify the name of the pattern you want to deploy. This deployment is for a **multiple rack only**.

There are several prerequisites:

- ▶ Your ID must be authorized to use the environment profile for multiple domains.
- ▶ Your login ID should also be authenticated by an LDAP server (not the local user repository).
- ▶ Ensure that your environment has been configured to enable multiple domain deployments.

To deploy the WebSphere Application Server cluster pattern, complete these steps:

1. Deploy the pattern.
 - a. Click the **Deploy** link for the pattern.
 - b. A new browser tab opens. Specify the following generic deployment parameters:
 - Name
 - Environment Profile (make sure to pick the environment profile that was created to support multiple racks). See “Environment profiles” on page 70 for details.
 - Priority
 - SSH key

- c. Specify the passwords for the root user, virtuser, and WebSphere admin user, see Figure B-7.

Figure B-7 Virtual system pattern deployment

- d. Click **Continue to Distribute**.
- e. The next window allows you to distribute different parts of the pattern as separate virtual machines on both the racks, as shown in Figure B-8.

| Components | 1000202 RedBookHADR External CG | 1000236 RedBooks HADR External network CG |
|------------|------------------------------------|--|
| IHSNode | 1 VM | |
| DmgrNode | 1 VM | |
| CustomNode | 1+ VM | 1+ VM |

Figure B-8 Distributing pattern parts

- f. Click **Deploy**.

- g. After the deployment completes, view the status in the **Instances** → **Virtual System Instances** window as shown in Figure B-9.

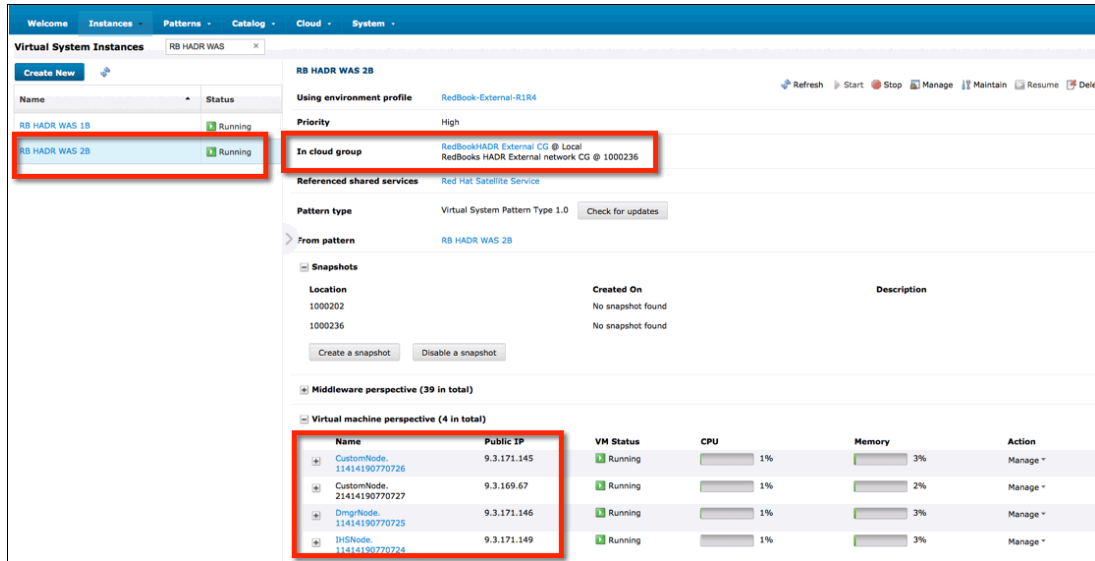


Figure B-9 Virtual system instances display

The pattern instance is spanning two cloud groups, one of which is on a different rack.

2. Verify that the pattern deployment was successful from WebSphere Application Server as shown in Figure B-10.
 - a. Log in to the WebSphere Admin Console on the Deployment Manager.
 - b. Go to **System Administration** → **Nodes**.

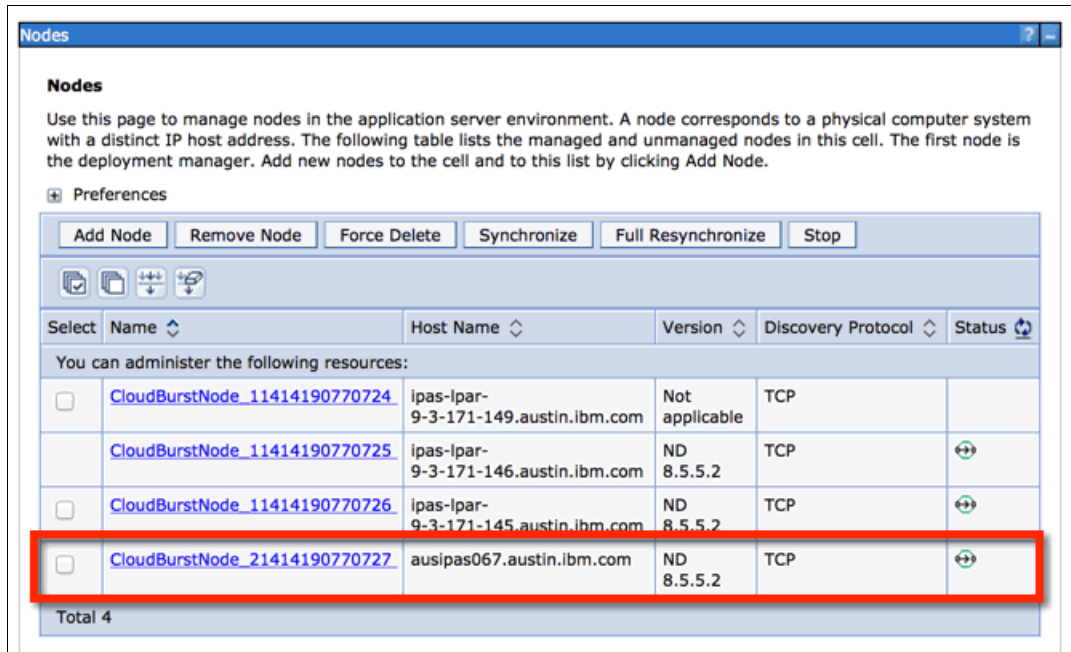


Figure B-10 WebSphere Application Server node management

One custom node virtual machine has a host name that is part of an IP group defined on the second rack. This shows that the WebSphere cell was created with one cell member on a different rack.

Create WebSphere Application Server cluster

The high-level steps to create a WebSphere Application Server cluster are outlined below.

1. Log in to the WebSphere Application Server administrative console. Use the IP address and host name of the Deployment Manager.
2. Create a cluster called *MyCluster* with a cluster member on each of the custom nodes that are part of the topology shown in Figure B-11.

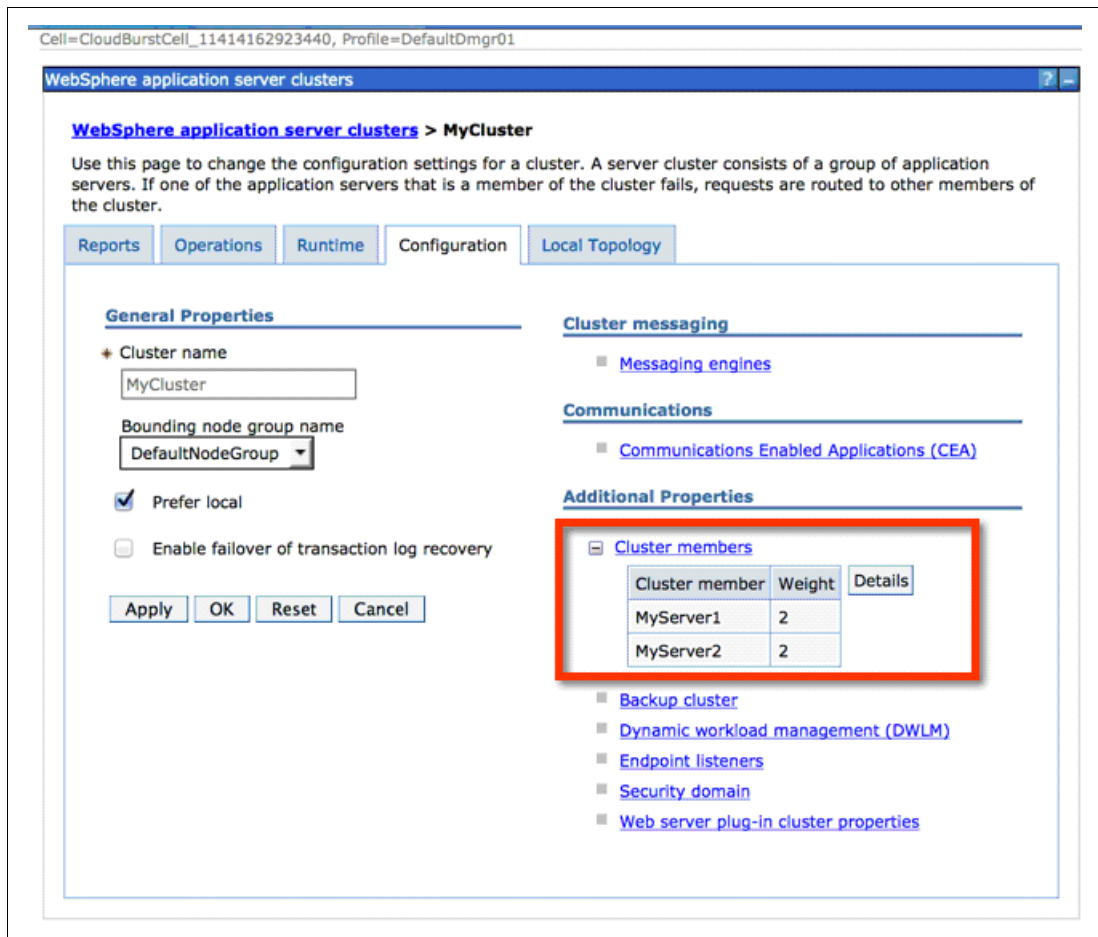


Figure B-11 Create a cluster

Configure database connectivity for BankTransaction application

Database configuration is discussed in “Database Setup” on page 255. In this section, database connectivity for the application is explained. This must be done before installing the application.

Log in to the WebSphere Application Server’s administration server as `virtuser`. Because the application requires two different data sources to connect to, update, and retrieve data, and create the appropriate JDBC provider to communicate with RDBMS.

The next sections provide information about how to complete the following requirements:

- ▶ “DB2 JDBC Database driver installation” on page 274
- ▶ “Create a JDBC Provider” on page 275
- ▶ “Create a J2C alias” on page 278
- ▶ “Create data sources for SAVINGS, CHECKING, and transaction data source” on page 280
- ▶ “Create data source for SAVINGS” on page 280

DB2 JDBC Database driver installation

Before creating a data source to connect the sample application to DB2, install the client DB2 database driver. This ensures that the application can connect to a database and run DB2 commands. The example in this chapter uses SSH to connect to the DB2 server instance as db2inst. To get started, first create the `opt/db2driver` directory and copy `db2jcc.jar` and `db2jcc_license_cu.jar` to the `opt/db2driver` directory.

To configure the JDBC database driver, log in to the WebSphere Application Server administrative console, and use the following steps:

1. Click the **WebSphere variables** link in the Environment section.
2. Click the **DB2UNIVERSAL_JDBC_DRIVER_PATH** link.
3. In the Value field, enter: `/opt/db2driver`.
4. Click **Apply**.

Figure B-12 shows the configuration for the JDBC Database driver.

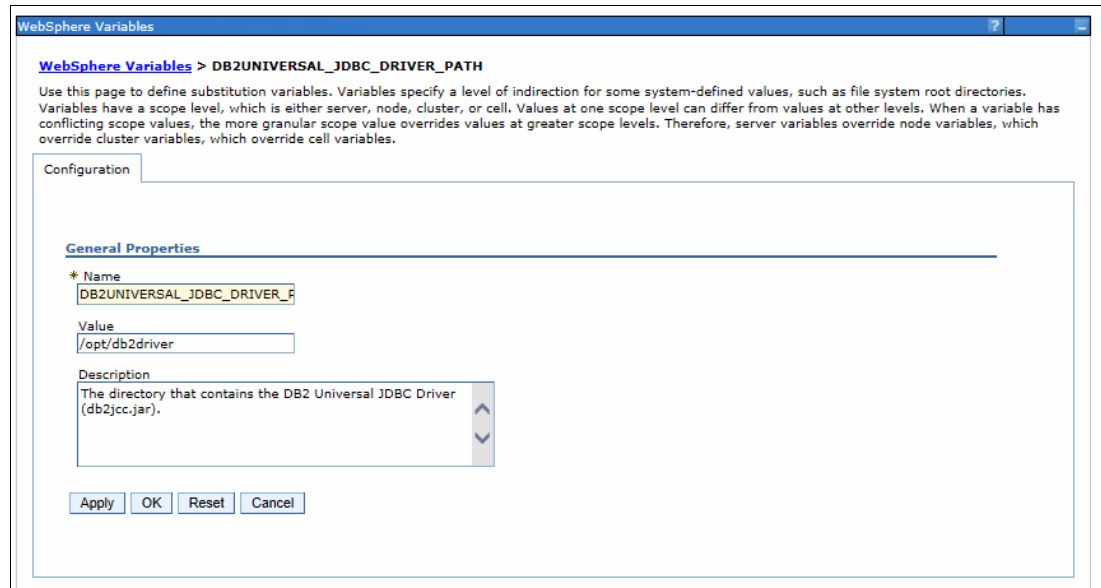


Figure B-12 Install DB2 JDBC client driver

Create a JDBC Provider

The application uses a JDBC provider that provides the actual implementation of the JDBC driver for access to DB2. Later, a data source is associated with a JDBC provider so that data source provides the connection to DB2. Figure B-13 shows the configuration setup for the JDBC Provider, and uses the following steps:

1. After logging in to the WebSphere Application Server administrative console, click the **JDBC Provider** link in the JDBC category of the Resources section
2. Select the scope as **cell** using the drop-down menu
3. Click **New**.

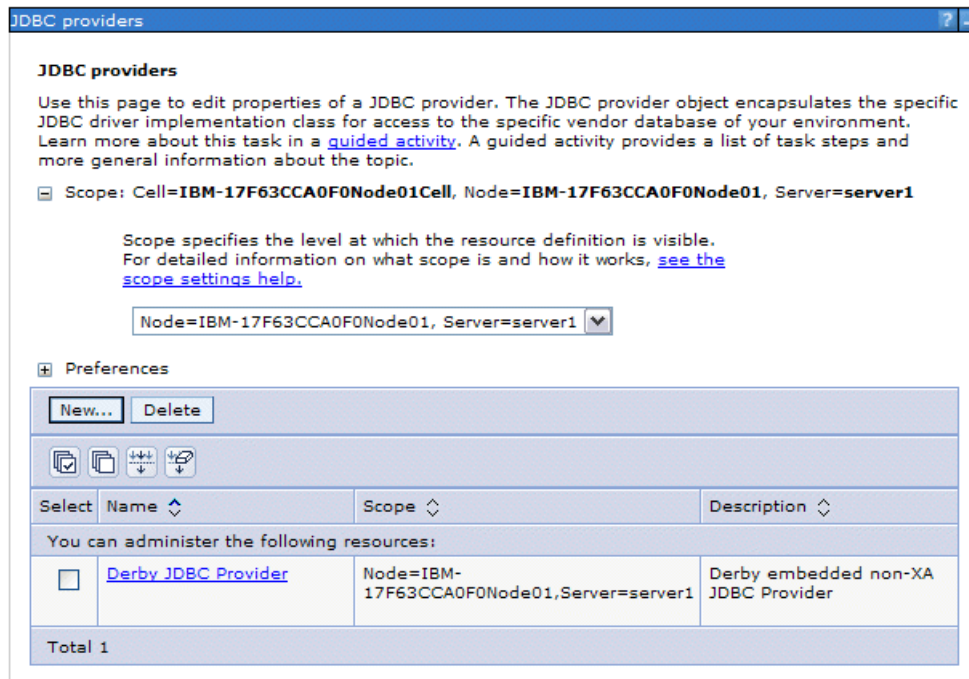


Figure B-13 Creating a JDBC provider

4. As shown in Figure B-14, enter these specifications:
 - a. Database type as **DB2**
 - b. Provider type as **DB2 Universal JDBC Driver Provider**
 - c. Implementation type as **XA data source** (because the application requires two-phase commit transactions)
 - d. Name is automatically filled as DB2 Universal JDBC Driver Provider (XA).
 - e. Click **Next**.

Note: Two-phase commit transactions allow a transaction to be rolled back during a failure. This process allows an application to handle the failure event.

Create a new JDBC Provider

→ Step 1: Create new JDBC provider

Step 2: Enter database class path information

Step 3: Summary

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.

Scope
cells:IBM-17F63CCA0F0Node01Cell:nodes:IBM-17F63CCA0F0Node01:servers:server1

* Database type
DB2

* Provider type
DB2 Universal JDBC Driver Provider

* Implementation type
XA data source

* Name
DB2 Universal JDBC Driver Provider (XA)

Description
Two-phase commit DB2 JCC provider that supports JDBC 3.0. Data sources that use this provider support the use of XA to perform 2-phase commit processing. Use of driver type 2 on the application server for z/OS is not supported for data sources created under this provider.

Next Cancel

Figure B-14 Create a JDBC provider with XA data source implementation type

- When the next page appears, review the information and click **Next** as shown in Figure B-15.

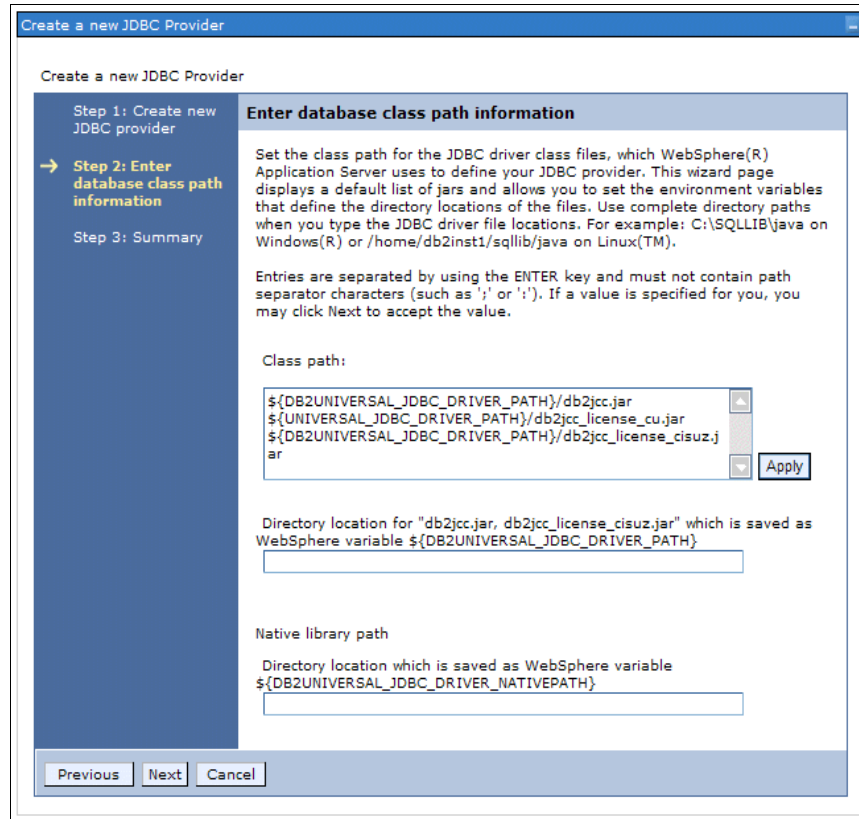


Figure B-15 Create a JDBC provider - database class path

- Click **Finish** as shown in Figure B-16.

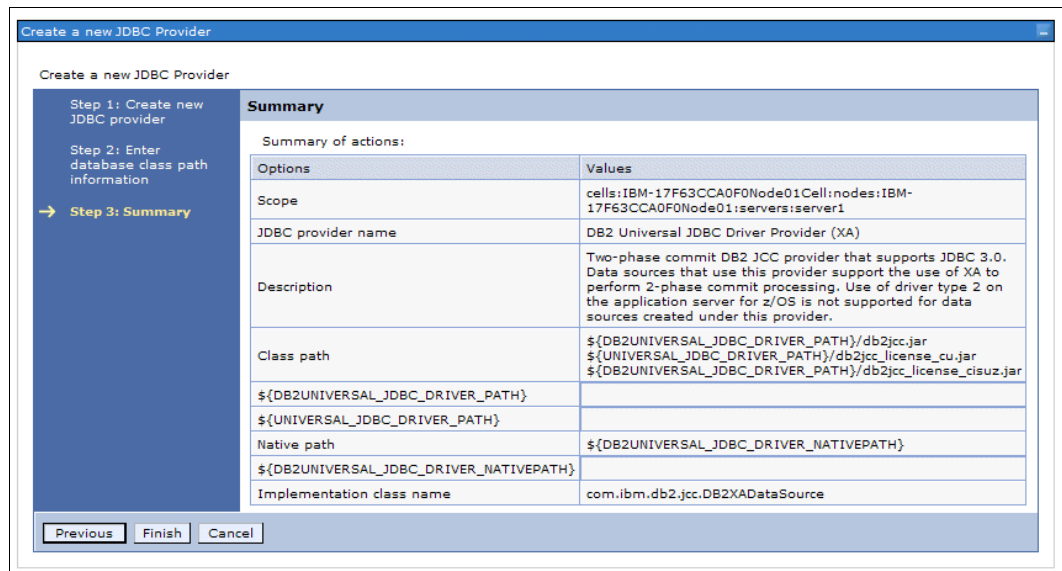


Figure B-16 Create a JDBC provider - summary

- Verify that the new JDBC Provider is created and save your results as shown in Figure B-17.

| Select | Name | Scope | Description |
|--------------------------|---|---|--|
| <input type="checkbox"/> | DB2 Universal JDBC Driver Provider (XA) | Node=IBM-17F63CCA0F0Node01,Server=server1 | Two-phase commit DB2 JCC provider that supports JDBC 3.0. Data sources that use this provider support the use of XA to perform 2-phase commit processing. Use of driver type 2 on the application server for z/OS is not supported for data sources created under this provider. |

Figure B-17 Create a JDBC provider- created JDBC provider

Create a J2C alias

Because the database has security enabled, it requires a user ID and password for the connection. To provide these requirements, you must create an authentication resource by creating a J2C alias using the following steps:

- Go to the Security section and click the **Global security** link.
- In the Authentication section, expand **Java Authentication and Authorization Service** as shown in Figure B-18.

Global security

Use this panel to configure administration and the default application security policy. This security configuration applies to the security administrative functions and is used as a default security policy for user applications. Security domains can be defined to override a security policies for user applications.

Security Configuration Wizard Security Configuration Report

Administrative security

- Enable administrative security
 - [Administrative user roles](#)
 - [Administrative group roles](#)
 - [Administrative authentication](#)

Application security

- Enable application security

Java 2 security

- Use Java 2 security to restrict application access to local resources
 - Warn if applications are granted custom permissions
 - Restrict access to resource authentication data

User account repository

Realm name:

Current realm definition:

Authentication

Authentication mechanisms and expiration

- [LTPA](#)
- Kerberos and LTPA
 - [Kerberos configuration](#)
- SWAM (deprecated): No authenticated communication
 - [Authentication cache settings](#)
- Web and SIP security
- RMI/IIOP security
- Java Authentication and Authorization Service**
 - [Application logins](#)
 - [System logins](#)
 - [J2C authentication data](#)
- Enable Java Authentication SPI (JASPI)
 - [Providers](#)
- Use realm-qualified user names

Figure B-18 Expand Java Authentication and Authorization Service section

- Click the **J2c authentication data** link, as shown in Figure B-19.

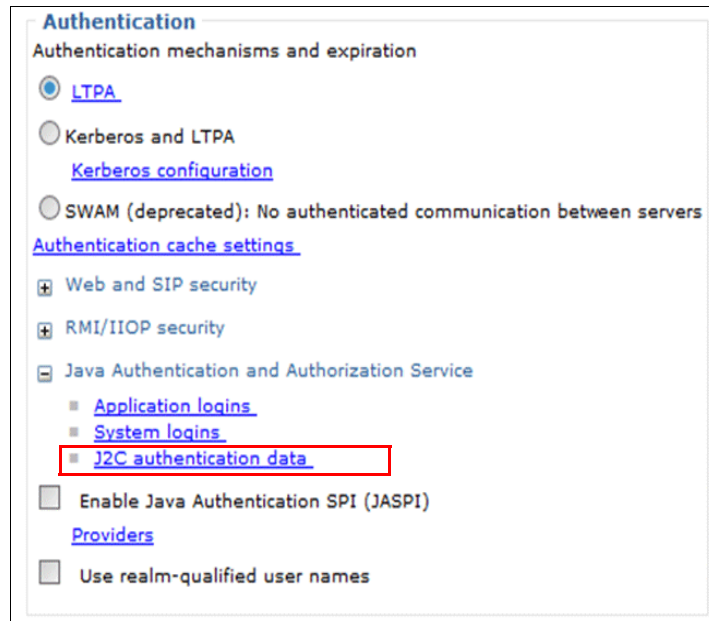


Figure B-19 J2C authentication data link

- Click **New** to create a J2C authentication alias.
- Enter SAVINGS for the alias, db2inst1 for the User ID, and the *password* for the DB2 pattern. The results of this configuration are shown in Figure B-20.

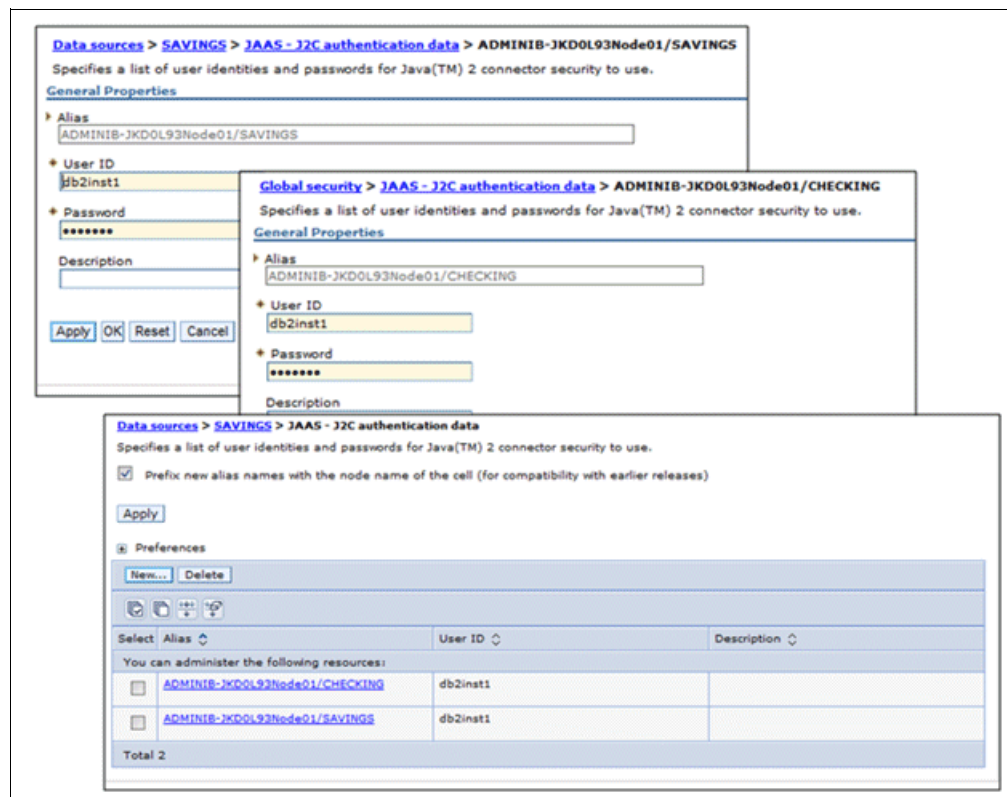


Figure B-20 Create J2c alias for SAVINGS and CHECKING

Note: Repeat steps 4 and 5 for the CHECKING alias.

Create data sources for SAVINGS, CHECKING, and transaction data source

For the application to use JNDI to access the SAVINGS database and the CHECKING database, you must create the JDBC data sources. When data sources are created, the J2C authentication aliases created in “Create a J2C alias” on page 278 are referred.

Create data source for SAVINGS

Use the following steps to create the data source for SAVINGS, as shown in Figure B-21:

1. Go to the JDBC section and click **Data sources**.
2. Select the **Cell scope** and click **New**.
3. Enter SAVINGS in the **Data source** name field, and the value jdbc/savings in the **JNDI name** field. The JNDI name provides a naming context for the data source as used in resource lookups by your application.
4. Click **Next**.

Create a data source

Create a data source

→ Step 1: Enter basic data source information

Step 2: Select JDBC provider

Step 3: Enter database specific properties for the data source

Step 4: Setup security aliases

Step 5: Summary

Enter basic data source information

Set the basic configuration values of a datasource for association with your JDBC provider. A datasource supplies the physical connections between the application server and the database.

Requirement: Use the Datasources (WebSphere(R) Application Server V4) console pages if your applications are based on the Enterprise JavaBeans(TM) (EJB) 1.0 specification or the Java(TM) Servlet 2.2 specification.

Scope
cells:ADMINIB-JKD0L93Node01Cell

* Data source name
SAVINGS

* JNDI name
jdbc/savings

Next Cancel

Figure B-21 Create data source for SAVINGS

- Click **Select an existing JDBC driver** and select **DB2 Universal JDBC Driver Provider (XA)**. This driver was created previously in “Create a JDBC Provider” on page 275. Then, click **Next** as shown in Figure B-22.

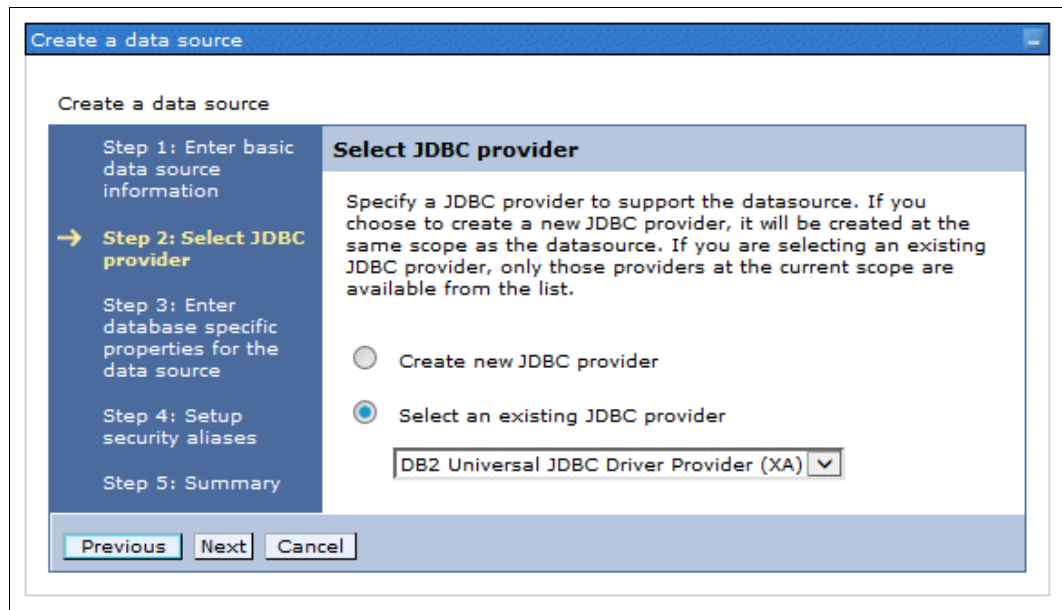


Figure B-22 Select JDBC provider

- Enter SAVINGS for the **Database name**. Then, enter your **Server name** and **Port number**, and click **Next**. See Figure B-23.

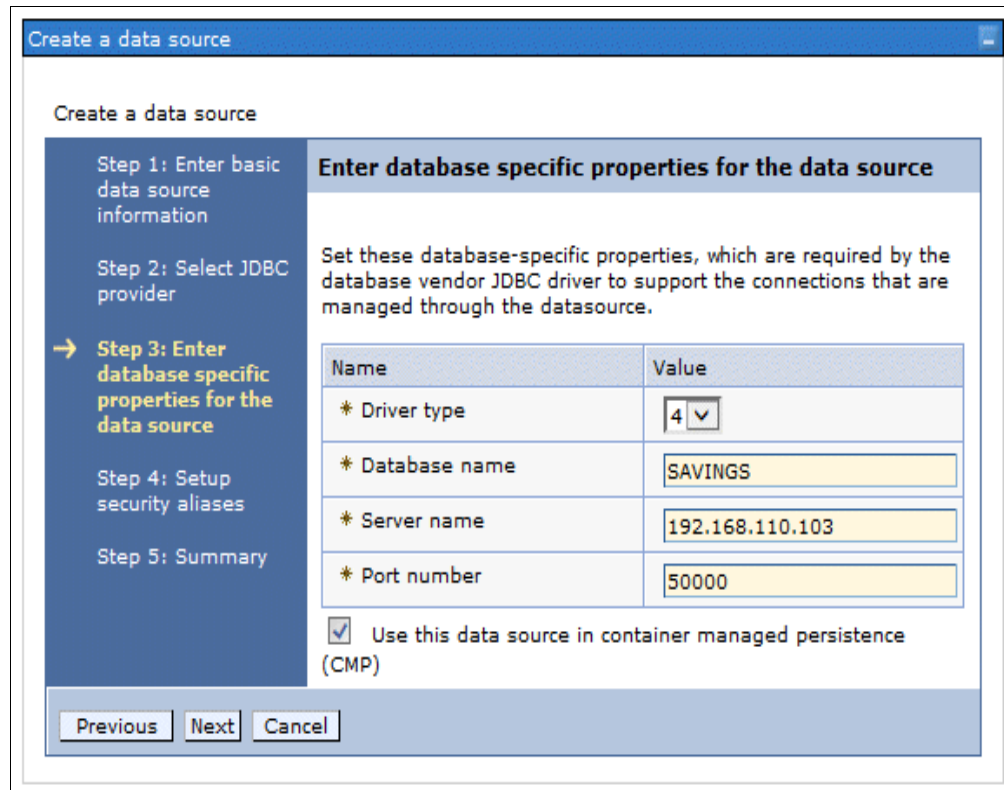


Figure B-23 Enter database-specific properties

- For all the security aliases, except mapping, from the drop-down menus, select J2C authentication alias **SAVINGS** and click **Next** as shown in Figure B-24. Do not select the mapping-configuration alias, which does not use a container-managed authentication alias.

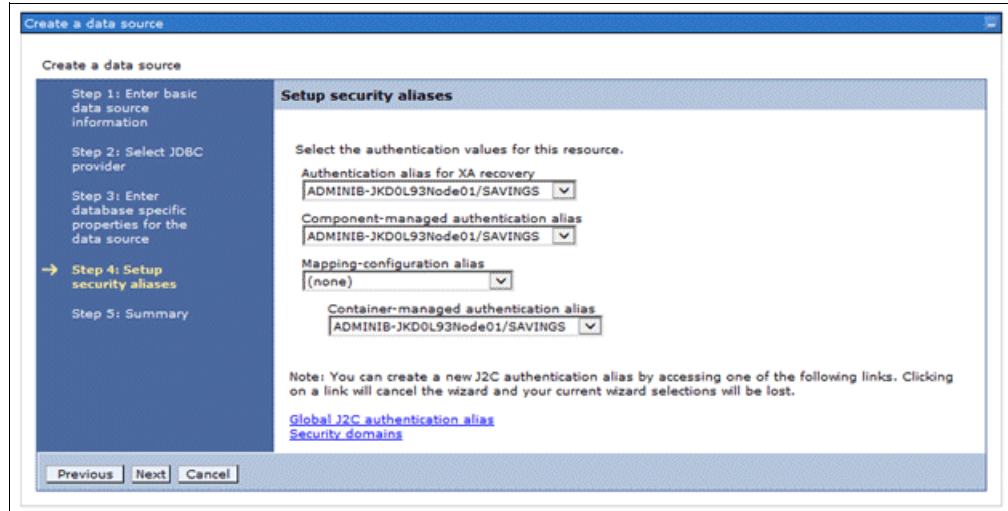


Figure B-24 Configuring J2C security alias

- View the summary window and click **Finish**, as shown in Figure B-25.

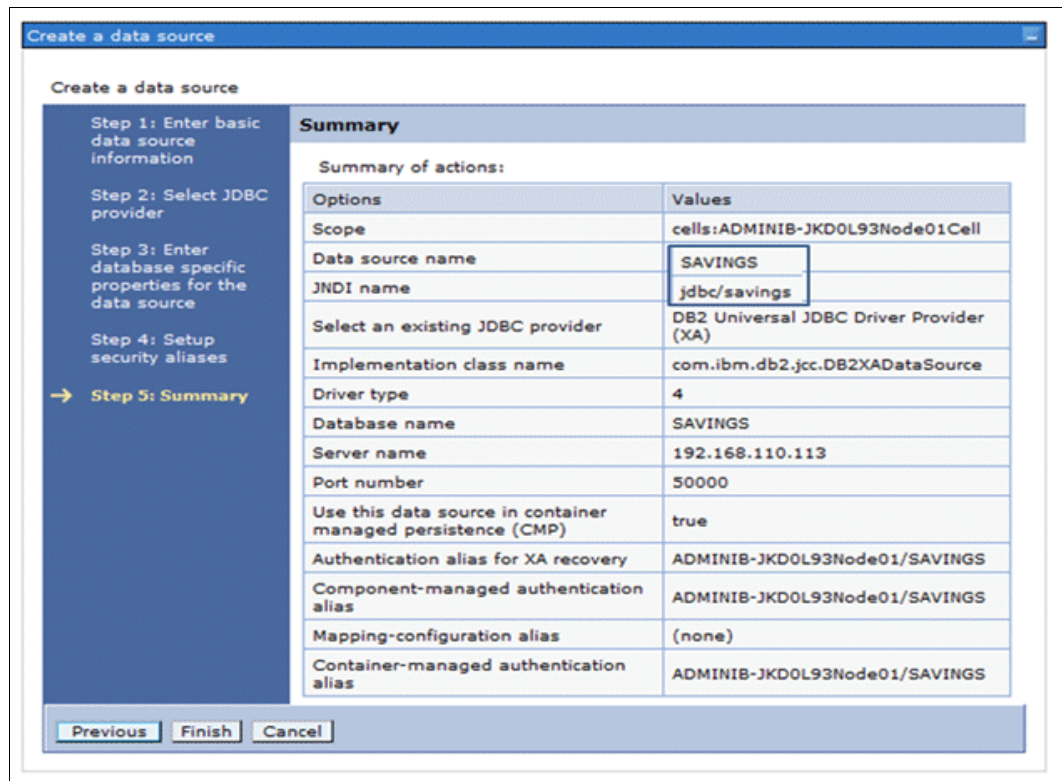


Figure B-25 Finalizing the data source for SAVINGS

- Repeat the process in “Create data source for SAVINGS” on page 280 to create another data source called CHECKING. The final data sources are shown in Figure B-26.

Data sources

Use this page to edit the settings of a datasource that is associated with your selected JDBC provider. The datasource object supplies your application with connections for accessing the database. Learn more about this task in a [guided activity](#). A guided activity provides a list of task steps and more general information about the topic.

[-] Scope: Cell=ADMINIB-JKD0L93Node01Cell

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

Cell=ADMINIB-JKD0L93Node01Cell

[+] Preferences

New... Delete Test connection Manage state...

| Select | Name | JNDI name | Scope | Provider | Description | Category |
|---|--------------------------|---------------|--------------------------------|---|---------------------------------|----------|
| You can administer the following resources: | | | | | | |
| <input type="checkbox"/> | CHECKING | jdbc/checking | Cell=ADMINIB-JKD0L93Node01Cell | DB2 Universal JDBC Driver Provider (XA) | DB2 Universal Driver Datasource | |
| <input type="checkbox"/> | SAVINGS | jdbc/savings | Cell=ADMINIB-JKD0L93Node01Cell | DB2 Universal JDBC Driver Provider (XA) | DB2 Universal Driver Datasource | |
| Total 2 | | | | | | |

Figure B-26 Data sources created for SAVINGS and CHECKING

- Check each data source and click **Test connection** to verify the connection to the target database.

Install BankTransaction application

The BankTransaction application is provided to help validate the scenarios in this publication. You can find functional details in Appendix A, “Sample Application” on page 253.

1. Install the application *BankTransaction* using the EAR. See Appendix C, “Additional material” on page 287 for instructions on downloading the sample application) as shown in Figure B-27.

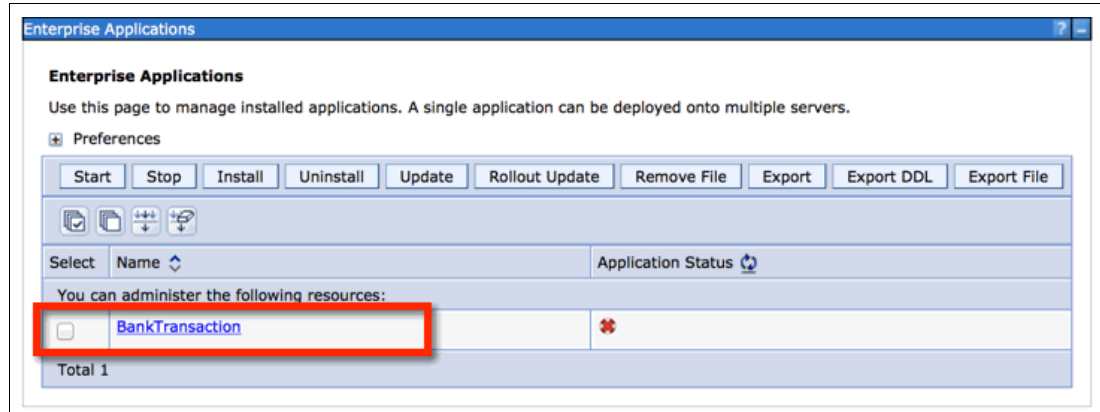


Figure B-27 Installing BankTransaction application

Note: To specify the option to use a database to store the transaction and compensation logs, follow the directions outlined at the following website. In these scenarios, two application servers are part of the cluster. Set this for each application server.

<http://www-01.ibm.com/support/docview.wss?uid=swg27038432>

2. Start the application servers from the administrative console.

Validate BankTransaction application

These instructions show you how to run several transactions and terminate (kill) one cluster member, either ClusterMember1 or ClusterMember2.

1. Validate that the application functions as shown in Figure B-28.
 - a. Start the bank transaction using the following URL:
`http://<IP/HostName>/BankTransactionWeb/banktransaction.html`

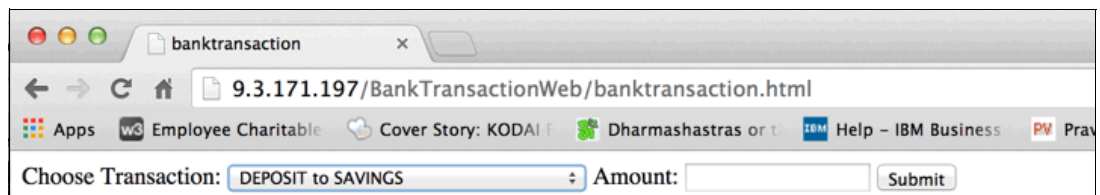


Figure B-28 Start the bank transaction

- b. From the drop-down, select a transaction that transfers from checking to savings and click **Submit**.

- c. There is a sleep configured in the application in between the two transactions of withdrawing from checking and depositing to savings.
- d. By looking at the SystemOut.log file, notice that the request was served as shown in Figure B-29.

```
[10/24/14 17:26:51:942 UTC] 00000092 InternalDB2Un I DSR48212I: DataStoreHelper name is: com.ibm.websphere.rsadapter.DB2Uni
versalDataStoreHelper@cc8db8a7.
[10/24/14 17:26:51:942 UTC] 00000092 WSRdbDataSour I DSR48208I: JDBC_driver type : 4
[10/24/14 17:46:14:855 UTC] 000000b1 SystemOut 0 Amount has been withdrawn from Checking ...now sleep 1 minute..
[10/24/14 17:46:45:908 UTC] 00000081 AdminHelper A ADMIN202I: An attempt is made to stop the myserver1 server. (user id =
defaultWIMFileBasedRealm/virtuser)
```

Figure B-29 SystemOut.log for withdrawal

- e. Terminate the application server using `kill -9`.
- f. Looking at the SystemOut.log file notice that the request has now been transferred to the other server because transaction recovery was configured using a database. See Figure B-30.

```
[10/24/14 17:50:16:916 UTC] 00000092 SystemOut 0 1 minute passed ...now wakeup to deposit to CHECKING..
[10/24/14 17:50:18:102 UTC] 0000003a DBUConnAdapter I DCSV1032I: DCS Stack DefaultCoreGroup at member CloudBurstCell
_11414162923440\CloudBurstNode_21414162923442\MyServer2: Connected a defined member CloudBurstCell_11414162923440\Clo
udBurstNode_11414162923441\MyServer1.
```

Figure B-30 SystemOut.log for deposit

2. The validation steps that are outlined in step 1 show how validation is handled on the WebSphere Application Server. Chapter 5, “High availability and disaster recovery scenarios for DB2” on page 89 discusses how the database itself can be failed over. After the database recovers, the application should behave normally.



Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG248246>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248246.

Using the web material

The additional web material that accompanies this book includes the following files:

- ▶ **BankTransaction.ear**: sample application used in scenarios
- ▶ **RB-HADR-WMQ_1.zip**: packaged code samples used in scenarios
- ▶ **RB-HADR-WMQ_2_Primary.zip**: packaged code samples used in scenarios
- ▶ **RB-HADR-WMQ_2_Standby.zip**: packaged code samples used in scenarios
- ▶ **WMQ-DelQMgr.zip**: packaged code samples used in scenarios
- ▶ **WMQExecuteMQSC.zip**: packaged code samples used in scenarios
- ▶ **WMQExecuteSingleMQSC.zip**: packaged code samples used in scenarios
- ▶ **WMQ-HAQMGR-Primary.zip**: packaged code samples used in scenarios
- ▶ **WMQ-HAQMGR-Standby.zip**: packaged code samples used in scenarios

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and download the contents of the web material files into this folder. Extract the SG248246.zip file into this folder to access the sample application and code sample files.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Always On: Assess, Design, Implement, and Manage Continuous Availability*, REDP-5109

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ For a list of supported SVC hardware, see the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extstorage_plan.dita?lang=en

- ▶ Supported hardware and procedures to enable the system to use external storage, refer to the IBM Knowledge Center article “Planning to use external storage”:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extstorage_plan.dita?lang=en

- ▶ Configuring external storage is found in the Knowledge Center in the article *Configuring the system to use external storage devices*:

http://www-01.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/systemconsole/extstorage_cfg.dita

- ▶ DB2 database:

http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/c0011724.html

- ▶ DB2 JCC JDBC Provider:

http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.apdv.java.doc/src/tpc/imjcc_c0056186.html

- ▶ Automatic Client Reroute:

<http://www.ibm.com/support/docview.wss?uid=swg21394840>

<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/DB2HADR/page/Client%20Reroute>

- ▶ How to export/import Virtual System Patterns:
http://www.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/iwd/pat_exportvsys.dita
http://www.ibm.com/support/knowledgecenter/SSCR9A_2.0.0/doc/iwd/pat_importvsys.dita
- ▶ WebSphere Application Server: Storing transaction and compensation logs in a relational database for high availability:
<http://www-01.ibm.com/support/docview.wss?uid=swg27038432>
- ▶ WebSphere MQIM:
https://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wmq_v7/wmq/7.0.1/Details/iea_701_120_multi_instancei.pdf?dmuid=20091218113354709936

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

Implementing High Availability and Disaster Recovery in IBM PureApplication Systems V2

(0.5" spine)
0.475" x 0.873"
250 x 459 pages



Implementing High Availability and Disaster Recovery in IBM PureApplication Systems V2



Discover new features in IBM PureApplication System V2.0

This IBM Redbooks publication describes and demonstrates common, prescriptive scenarios for setting up disaster recovery for common workloads using IBM WebSphere Application Server, IBM DB2, and WebSphere MQ between two IBM PureApplication System racks using the features in PureApplication System V2.

Examine how to use GPFS with WebSphere Application Server

The intended audience for this book is pattern developers and operations team members who are setting up production systems using software patterns from IBM that must be highly available or able to recover from a disaster (defined as the complete loss of a data center).

Learn about Block Storage Replication

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

SG24-8246-00

ISBN 0738440337