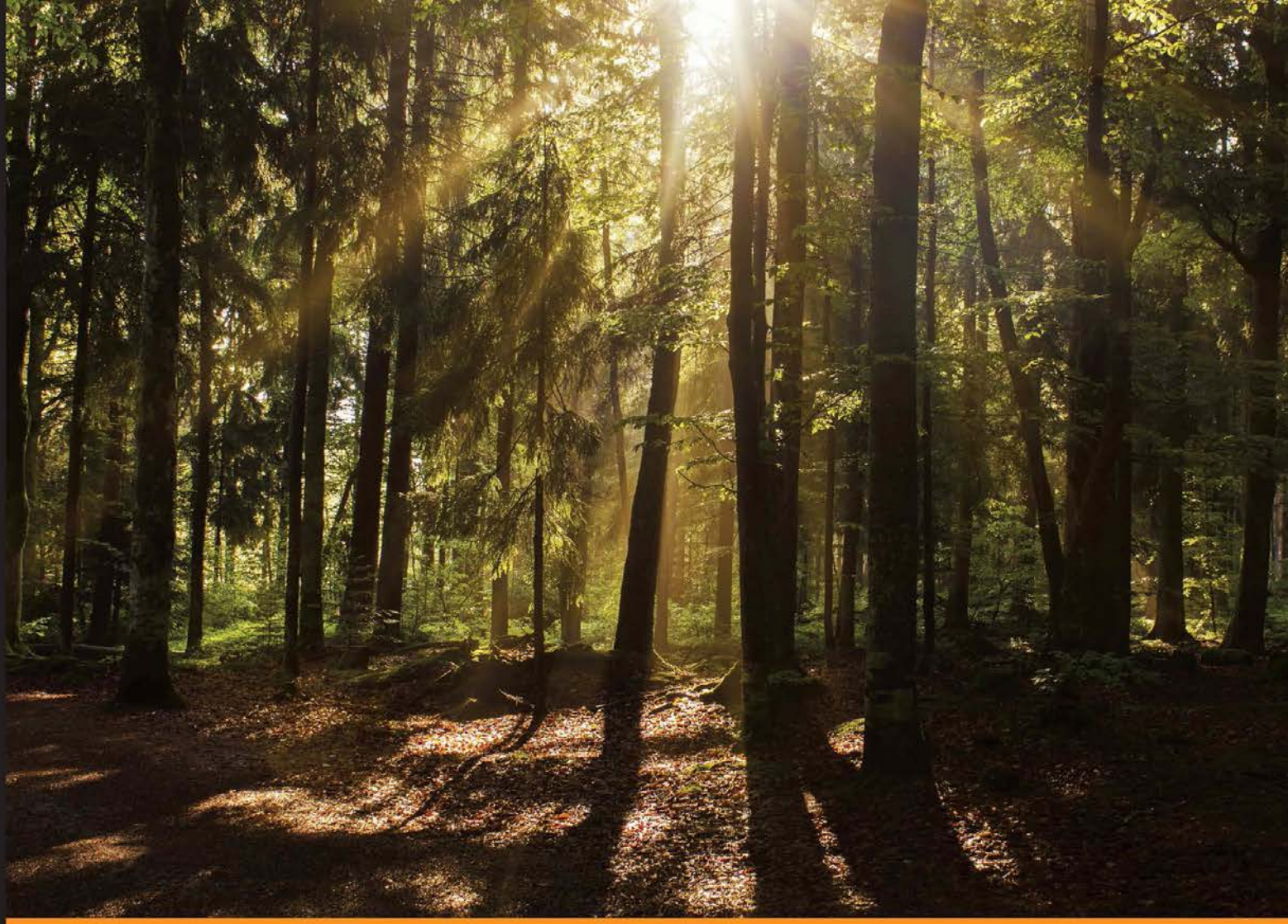


Community Experience Distilled

Learning RabbitMQ

Build and optimize efficient messaging applications with ease

Martin Toshev

[PACKT] PUBLISHING open source* community experience distilled

Learning RabbitMQ

Build and optimize efficient messaging applications with ease

Martin Toshev

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

Learning RabbitMQ

First published: December 2015

Production reference: 1171215

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78398-456-5

www.packtpub.com

Credits

Author
Martin Toshev

Reviewers
Van Thoai Nguyen
Héctor Veiga

Commissioning Editor
Ashwin Nair

Acquisition Editor
Vinay Argekar

Content Development Editor
Kirti Patil

Technical Editor
Danish Shaikh

Copy Editor
Vibha Shukla

Project Coordinator
Nidhi Joshi

Proofreader
Safis Editing

Indexer
Hemangini Bari

Graphics
Disha Haria

Production Coordinator
Arvindkumar Gupta

Cover Work
Arvindkumar Gupta

About the Author

Martin Toshev is a software developer and Java enthusiast with more than eight years of experience and vast expertise originating from projects in areas such as enterprise Java, social networking, source code analysis, Internet of Things, and investment banking in companies such as Cisco and Deutsche Telekom. He is a graduate of computer science from the University of Sofia. He is also a certified Java professional (SCJP6) and a certified IBM cloud computing solution advisor. His areas of interest include a wide range of Java-related technologies (Servlets, JSP, JAXB, JAXP, JMS, JMX, JAX-RS, JAX-WS, Hibernate, Spring Framework, Liferay Portal, and Eclipse RCP), cloud computing technologies, cloud-based software architectures, enterprise application integration, and relational and NoSQL databases. Martin is one of the leaders of the Bulgarian Java Users group (BGJUG), a regular speaker at Java conferences, and one of the organizers behind the jPrime conference in Bulgaria (`http://jprime.io/`).

About the Reviewers

Van Thoai Nguyen has worked in the software industry for a decade in various domains. In 2012, he joined BuzzNumbers as one of the core senior software engineers, where he had opportunities to design, implement, and apply many cool technologies, tools, and frameworks. A RabbitMQ cluster was employed as the backbone of the real-time data processing platform, which includes various data collectors, data filtering, enrichment, and storage using a sharded cluster of MongoDB and SOLR. He is still maintaining the open source .NET RabbitMQ client library, Burrow.NET (`https://github.com/vanthoainguyen/Burrow.NET`), which he built during the time he worked for BuzzNumbers. This library is still being used in many different applications in that company. Van is interested in clean code and design, SOLID principle, and BIG data. You can read his blog at `http://thoai-nguyen.blogspot.com.au/`.

Héctor Veiga is a software engineer specializing in real-time data integration and processing. Recently, he has focused his work on different cloud technologies, such as AWS, to develop scalable, resilient, and high-performing applications with the latest open source technologies, such as Scala, Akka, or Apache Spark. Additionally, he has a strong foundation in messaging systems, such as RabbitMQ and AMQP. He also has a master's degree in telecommunications engineering from the Universidad Politécnica de Madrid and a master's degree in information technology and management from the Illinois Institute of Technology.

He currently works as part of the Connected Driving real-time data collection team and is actively developing scalable applications to ingest and process data from several different sources. He utilizes RabbitMQ heavily to address their messaging requirements. In the past, he worked at Xaptum Technologies, a company dedicated to M2M technologies.

Héctor also helped with the reviewing process of *RabbitMQ Cookbook* and *RabbitMQ Essentials*, both from *Packt Publishing*.

I would like to thank my parents, Pilar and Jose Carlos, as well as my sister, Paula, for always supporting me and motivating me to keep pushing on. Without them, all this would not have been possible.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and readPackt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

I would like to thank all of the people that supported me during the process of writing this book and especially my mother Milena, my beloved Tsveti and my grandmother Maria. Without them this would not have been possible.

Table of Contents

Preface vii

Chapter 1: Introducing RabbitMQ 1

- **Enterprise messaging** 2
  - Use cases 6
  - Solutions 7
  - Patterns 7
    - Point-to-point 7
    - Publish-subscribe 7
    - Request-response 7
- **Understanding RabbitMQ** 8
  - Features 10
  - Comparison with other technologies 11
  - Installation 11
    - Linux 16
- **Case study: CSN (Corporate Social Network)** 17
- **Summary** 18
- **Exercises** 18

Chapter 2: Design Patterns with RabbitMQ 19

- **Messaging patterns in RabbitMQ** 19
- **Point-to-point communication** 23
- **Publish-subscribe communication** 29
- **Request-reply communication** 34
- **Message router** 39
- **Case study: Initial design of the CSN** 41
- **Summary** 42
- **Exercises** 42

Chapter 3: Administration, Configuration, and Management 43
Administering RabbitMQ instances 43
Administering RabbitMQ components 46
Administering users 47
Administering vhosts 49
Administering permissions 50
Administering exchanges 50
Administering queues 51
Administering bindings 52
Administering policies 52
Administering the RabbitMQ database 55
Full backup and restore 55
Backing up and restoring the broker metadata 56
Installing RabbitMQ plugins 57
Configuring RabbitMQ instances 58
Setting environment variables 58
Modifying the RabbitMQ configuration file 59
Managing RabbitMQ instances 59
Upgrading RabbitMQ 62
Case study: Administering CSN 63
Summary 64
Exercises 65
Chapter 4: Clustering 67
Benefits of clustering 67
RabbitMQ clustering support 68
Creating a simple cluster 69
Adding nodes to the cluster 70
Adding RAM-only nodes to the cluster 73
Removing nodes from a cluster 74
Connecting to the cluster 75
Case study: scaling the CSN 81
Summary 82
Exercises 82
Chapter 5: High Availability 83
Benefits of high availability 84
High availability support in RabbitMQ 85
Mirrored queues 86
Federation plugin 90
Shovel plugin 96

Reliable delivery 98
AMQP transactions 100
Publisher confirms 103
Client high availability 104
Client reconnections 104
Load balancing 104
Case study: introducing high availability in CSN 105
Summary 106
Exercises 106
Chapter 6: Integrations 107
Types of integrations 108
Spring framework 109
Spring AMQP 110
Spring Integration 113
Integration with ESBs 116
Mule ESB 116
WSO2 122
Integration with databases 127
Oracle RDBMS 127
MongoDB 129
Hadoop 129
RabbitMQ integrations 130
RabbitMQ deployment options 130
Puppet 131
Docker 132
Vagrant 133
Testing RabbitMQ applications 133
Unit testing of RabbitMQ applications 133
Integration testing of RabbitMQ applications 133
Case study: Integrating CSN with external systems 134
Summary 135
Exercises 135
Chapter 7: Performance Tuning and Monitoring 137
Performance tuning of RabbitMQ instances 137
Memory usage 139
Faster runtime execution 140
Message size 141
The maximum frame size of messages 141
The maximum number of channels 141
Connection heartbeats 142
Clustering and high availability 142
QoS prefetching 143

Message persistence 144
Mnesia transaction logs 145
Acknowledgements, transactions and publisher confirms 145
Message routing 145
Queue creation/deletion 145
Queue message TTL 146
Alarms 147
Network tuning 148
Client tuning 149
Performance testing 149
Monitoring of RabbitMQ instances 154
The management UI 154
Nagios 156
Monit 158
Munin 159
Comparing RabbitMQ with other message brokers 162
Case Study : Performance tuning and monitoring of RabbitMQ instances in CSN 162
Summary 162
Exercises 163
Chapter 8: Troubleshooting 165
General troubleshooting approach 165
Checking the status of a particular node 166
Inspecting the RabbitMQ logs 167
The RabbitMQ mailing list and IRC channel 169
Erlang troubleshooting 169
An Erlang Primer 169
The Erlang crash dump 177
Problems with starting/stopping RabbitMQ nodes 179
Problems with message delivery 182
Summary 182
Exercises 183
Chapter 9: Security 185
Types of threats 185
Authentication 188
Configuring the LDAP backend 190
Security considerations 194
Authorization 194
LDAP authentication 195

Secure communication **198**
Secure communication with the management interface 201
Secure cluster communication 203
EXTERNAL SSL authentication 203
Penetration testing **203**
Case study – securing CSN **204**
Summary **206**
Exercises **206**
Chapter 10: Internals **207**
High level architecture of RabbitMQ **207**
Overview of RabbitMQ components **212**
Boot component 212
Plug-in loader component 215
Recovery component 216
Persistence component 217
Metadata persistence 217
Message persistence component 218
Networking component 219
Other components 220
Developing plug-ins for RabbitMQ 221
Case Study: Developing a RabbitMQ plugin for CSN **222**
Summary **222**
Exercises **223**
Appendix: Contributing to RabbitMQ **225**
RabbitMQ community **225**
RabbitMQ repositories **225**
Getting the sources 226
Building the RabbitMQ server 226
Points for contribution **230**
Index **231**

Preface

Learning RabbitMQ provides you with a practical guide for the notorious message broker and covers the essentials required to start using it. The reader is able to build up knowledge along the way—starting from the very basics (such as what is RabbitMQ and what features does it provide) and reaching the point where more advanced topics, such as RabbitMQ troubleshooting and internals, are discussed. Best practices and important tips are provided in a variety of scenarios; some of them are related to external systems that provide integration with the message broker or that are integrated as part of the message broker in the form of a RabbitMQ plugin. Practical examples are also provided for most of these scenarios that can be applied in a broader context and used as a good starting point.

An example system called **CSN** (**Corporate Social Network**) is used to illustrate the various concepts provided throughout the chapters.

Each chapter ends with an Exercises section that allows the reader to test his understanding on the presented topic.

What this book covers

Chapter 1, *Introducing RabbitMQ*, provides you with a brief recap on enterprise messaging and a short overview of RabbitMQ along with its features. Other similar technologies are mentioned and an installation guide for the message broker is provided at the end of the chapter. The basic terminology behind RabbitMQ such as exchanges, queues, and bindings is introduced.

Chapter 2, *Design Patterns with RabbitMQ*, discusses what messaging patterns can be implemented using RabbitMQ, including point-to-point, publish-subscribe, request-reply, and message router types of communication. The patterns are implemented using the building blocks provided by the message broker and using the Java client API.

Chapter 3, *Administration, Configuration and Management*, reveals how to administer and configure RabbitMQ instances, how to install and manage RabbitMQ plugins, and how to use the various utilities provided as part of the RabbitMQ installation in order to accomplish a number of administrative tasks. A brief overview of the RabbitMQ management HTTP API is provided.

Chapter 4, *Clustering*, discusses what built-in clustering support is provided in the message broker and how it can be used to enable scalability in terms of message queues. A sample RabbitMQ cluster is created in order to demonstrate how nodes can be added/removed from a cluster and how RabbitMQ clients can connect to the cluster.

Chapter 5, *High Availability*, extends on the concepts of clustering by providing an overview of how a RabbitMQ cluster can be made more reliable in terms of mirrored queues and how messages can be replicated between remote instances using the Federation and Shovel plugins. High availability in terms of client connections and reliable delivery is also discussed with AMQP transactions, publisher confirms, and client reconnections.

Chapter 6, *Integrations*, provides you with a number of practical scenarios for integration of the message broker with the Spring framework, with ESB (enterprise services bus) systems such as MuleESB and WS02, and with database management systems (RDBMS and NoSQL). Deployment options for RabbitMQ using systems such as Puppet, Docker, and Vagrant are discussed in the chapter. A brief overview of how RabbitMQ applications can be tested using third-party frameworks is provided at the end of the chapter.

Chapter 7, *Performance Monitoring and Tuning*, gives a detailed list of factors that must be considered in terms of performance tuning of the message broker. The PerfTest tool is used to demonstrate how the RabbitMQ performance can be tested. At the end of the chapter, several monitoring solutions that provide support for RabbitMQ such as Nagios, Munin, and Monit are used to demonstrate how the message broker can be monitored in terms of stability and performance.

Chapter 8, *Troubleshooting*, illustrates a number of problems that can occur during the startup of the message broker and normal operation along with the various causes and resolutions in such cases. A brief primer on the Erlang programming language is provided for the purpose of understanding and analyzing the RabbitMQ crash dump—either directly or using the Crashdump Viewer for convenience.

Chapter 9, *Security*, provides a high-level overview of the vulnerability landscape related to the message broker along with a number of techniques to secure a RabbitMQ setup. Authentication, authorization, and secure communication are among the most important concepts covered in the chapter.

Chapter 10, *Internals*, discusses the internal architecture of the message broker and provides a detailed overview on the most important components that RabbitMQ comprises of.

Appendix A, *Contributing to RabbitMQ*, provides a short guide on how to get the RabbitMQ sources, how to set up a development environment, and how to build the message broker. A short discussion on how to contribute to the RabbitMQ ecosystem is provided as part of the appendix.

What you need for this book

In order to get the most out of this book, the reader is expected to have at least a basic understanding of what messaging is all about and a good understanding in at least one object-oriented programming language. As the book features the RabbitMQ Java client API in order to demonstrate how to use the message broker, it is good to have at least a basic understanding of the Java programming language. Most of the examples are not specific to a particular operating system; if they are, it is explicitly mentioned whether this is, for example, a Windows- or Unix-based distribution such as Ubuntu. For this reason, there is no particular requirement for an operating system in order to run the examples.

Who this book is for

If you are a developer or system administrator with basic knowledge in messaging who wants to learn RabbitMQ or further enhance your knowledge in working with the message broker, then this book is ideal for you. For a full understanding of some the examples in the book, basic knowledge of the Java programming language is required. Feeling comfortable with RabbitMQ is a great way to leverage your expertise in the world of messaging systems.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, names of third-party applications, utilities, folder names, filenames, file extensions andpathnames are shown in bold as follows: "We already saw how easy it is to `start/stop/restart` instances using the `rabbitmqctl` and `rabbitmq-server` utilities that are part of the standard RabbitMQ installation."

A block of code displayed in a box with console font:

```
<dependency>
 <groupId>log4j</groupId>
 <artifactId>log4j</artifactId>
 <version>1.2.16</version>
</dependency>
```

A block of configuration or output is also displayed in a box as follows:

```
sudo apt-get install rabbitmq-server -y
sudo rabbitmq-plugins enable rabbitmq_management
sudo service rabbitmq-server restart
```

New terms and **important words** are also shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Introducing RabbitMQ

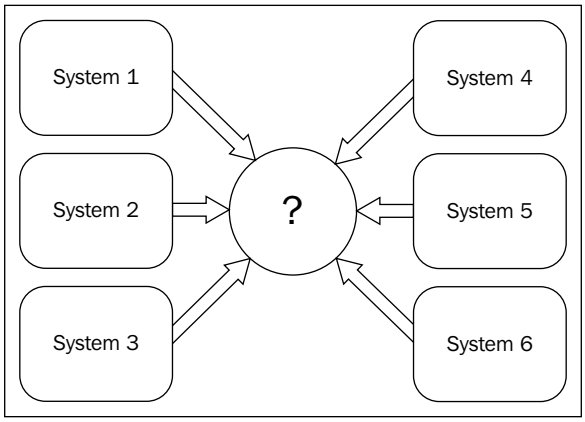
In the world of enterprise messaging systems there are a number of patterns and practices that are already successfully applied in order improve to scalability and interoperability between different components in a system or between varying in size and complexity systems. RabbitMQ is one such messaging solution, which combines powerful messaging capabilities with easy use and availability on a number of target platforms.

The following topics will be covered in this chapter:

- Fundamentals of enterprise messaging
- RabbitMQ brief overview
- RabbitMQ features
- Comparing RabbitMQ to other technologies
- Installing RabbitMQ

Enterprise messaging

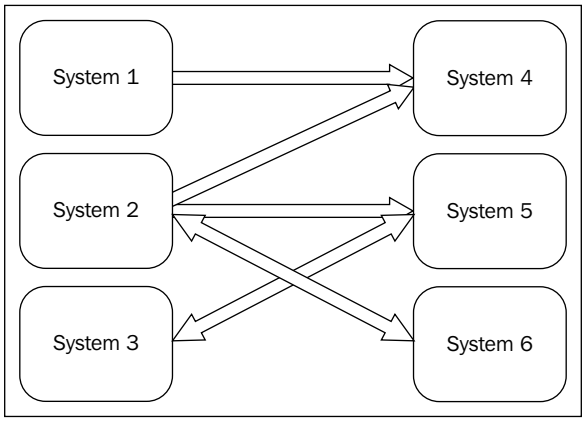
A typical enterprise will have a number of systems that must typically communicate with each other in order to implement a well-defined business process. A question that is frequently tackled for this reason is how to implement the communication channel between these types of systems? For example, consider the following diagram:



The question mark in the preceding picture denotes the communication media for the six systems that are illustrated. In the diagram, we can think of these separate systems as the components of one large system and the problem stays the same. Before discussing the various alternatives for integration, a number of key factors are considered, as follows:

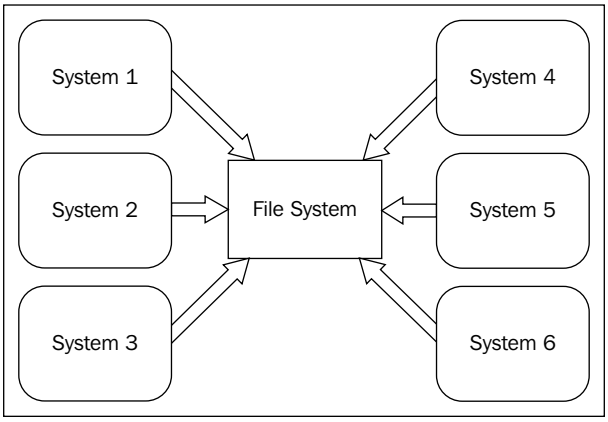
- **Loose coupling**: At what degree do the different systems depend on each other or can operate independently?
- **Real-time workload processing**: How fast is the communication between the systems?
- **Scalability**: How does the entire system scale when more systems are added and the workload demands an increase?
- **Maintainability**: How hard it is to maintain the integrated systems?
- **Extensibility**: How easy it is to integrate new systems?

Let's assume that the systems communicate directly via some kind of remote procedure calls as shown in the following diagram:



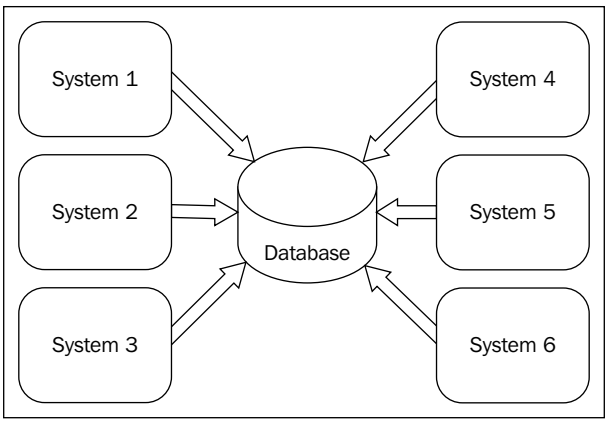
This implies that separate communication links must be established between each pair of systems, which leads to tight coupling, a lot of effort to maintain all of the links, reduced scalability, and reduced extensibility (for every new system that is added, a few more communication links with other systems must be created). However, real-time communication requirements might be met with some additional effort to design the communication links.

A second approach is to use a shared file system in order to exchange files between the systems that are being integrated, as illustrated in the following diagram:



A shared file system is used to provide the communication medium. Each system may export data to a file that can be imported and used by other systems. The fact that each system may support its own data format leads to the fact that each system must have a particular mechanism to import data from every other system that it needs to communicate with. This, on other hand, leads to the same problems that are described in the case of direct communication. Moreover, real-time communication requirements might be more difficult to establish and reading or writing data from disk is also an expensive operation.

A third option is to use a shared database as shown in the following diagram:



Here, all the systems should depend on the same database schema. Although this reduces coupling between systems and improves extensibility (new systems must conform to a single database schema in order to integrate with other systems), real-time workload processing is still an issue. Scalability and maintainability depend directly on the type of database that is being used and they could turn out to be weak factors especially if it is a relational database (this may not be the case if NoSQL solutions are applicable for the particular use case).