

O'REILLY®

# Kubernetes Patterns

Reusable Elements for Designing  
Cloud-Native Applications

*Grayscale Edition*

**For Sale in  
the Indian  
Subcontinent &  
Select Countries  
Only\***

\*Refer Back Cover



Bilgin Ibryam &  
Roland Huß

---

# Kubernetes Patterns

*Reusable Elements for Designing  
Cloud-Native Applications*

*Bilgin Ibryam and Roland Huß*

Beijing • Boston • Farnham • Sebastopol • Tokyo

**O'REILLY®**



**SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.**

*Mumbai*

*Bangalore*

*Kolkata*

*New Delhi*

# Kubernetes Patterns

by Bilgin Ibryam and Roland Huß

Copyright © 2019 Bilgin Ibryam and Roland Huß. All rights reserved. ISBN: 978-1-492-05028-5  
Originally printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://safari.oreilly.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

**Acquisitions Editor:** John Devins

**Developmental Editors:** Virginia Wilson

**Production Editor:** Katherine Tozer

**Copyeditor:** Christine Edwards

**Proofreader:** Sharon Wilkey

**Indexer:** Judith McConville

**Interior Designer:** David Futato

**Cover Designer:** Karen Montgomery

**Illustrator:** Rebecca Demarest

## Printing History:

May 2019: First Edition

## Revision History for the First Edition

 2019-04-04: First Release

See <https://www.oreilly.com/catalog/errata.csp?isbn=9781492050285> for release details.

## First Indian Reprint: April 2019

ISBN: 978-93-5213-826-5

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Kubernetes Patterns*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the authors, and do not represent the publisher's views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

**For sale in the Indian Subcontinent (India, Pakistan, Bangladesh, Sri Lanka, Nepal, Bhutan, Maldives) and African Continent (excluding Morocco, Algeria, Tunisia, Libya, Egypt, and the Republic of South Africa) only. Illegal for sale outside of these countries.**

Authorized reprint of the original work published by O'Reilly Media, Inc. All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, nor exported to any countries other than ones mentioned above without the written permission of the copyright owner.

Published by **Shroff Publishers & Distributors Pvt. Ltd.** B-103, Railway Commercial Complex, Sector 3, Sanpada (E), Navi Mumbai 400705 • TEL: (91 22) 4158 4158 • FAX: (91 22) 4158 4141 E-mail: [sporders@shroffpublishers.com](mailto:sporders@shroffpublishers.com) • Web: [www.shroffpublishers.com](http://www.shroffpublishers.com) Printed at Jasmine Art Printers Pvt. Ltd., Mumbai.

---

# Table of Contents

<b>Foreword.....</b>	<b>ix</b>
<b>Preface.....</b>	<b>xi</b>
<b>1. Introduction.....</b>	<b>1</b>
The Path to Cloud Native	1
Distributed Primitives	3
Containers	4
Pods	5
Services	7
Labels	7
Annotations	9
Namespaces	9
Discussion	11
More Information	12

---

## Part I. Foundational Patterns

<b>2. Predictable Demands.....</b>	<b>15</b>
Problem	15
Solution	16
Runtime Dependencies	16
Resource Profiles	18
Pod Priority	20
Project Resources	22
Capacity Planning	22

Discussion	23
More Information	24
<b>3. Declarative Deployment.....</b>	<b>25</b>
Problem	25
Solution	25
Rolling Deployment	27
Fixed Deployment	29
Blue-Green Release	30
Canary Release	30
Discussion	31
More Information	33
<b>4. Health Probe.....</b>	<b>35</b>
Problem	35
Solution	35
Process Health Checks	36
Liveness Probes	36
Readiness Probes	37
Discussion	38
More Information	40
<b>5. Managed Lifecycle.....</b>	<b>41</b>
Problem	41
Solution	41
SIGTERM Signal	42
SIGKILL Signal	42
Poststart Hook	43
Prestop Hook	44
Other Lifecycle Controls	45
Discussion	46
More Information	46
<b>6. Automated Placement.....</b>	<b>47</b>
Problem	47
Solution	47
Available Node Resources	48
Container Resource Demands	49
Placement Policies	49
Scheduling Process	50
Node Affinity	51

Pod Affinity and Antiaffinity	52
Taints and Tolerations	54
Discussion	57
More Information	59

---

## Part II. Behavioral Patterns

<b>7. Batch Job</b> .....	<b>63</b>
Problem	63
Solution	64
Discussion	67
More Information	68
<b>8. Periodic Job</b> .....	<b>69</b>
Problem	69
Solution	70
Discussion	71
More Information	72
<b>9. Daemon Service</b> .....	<b>73</b>
Problem	73
Solution	74
Discussion	76
More Information	77
<b>10. Singleton Service</b> .....	<b>79</b>
Problem	79
Solution	80
Out-of-Application Locking	80
In-Application Locking	82
Pod Disruption Budget	84
Discussion	85
More Information	86
<b>11. Stateful Service</b> .....	<b>87</b>
Problem	87
Storage	88
Networking	89
Identity	89
Ordinality	89

Other Requirements	89
Solution	90
Storage	91
Networking	92
Identity	94
Ordinality	94
Other Features	95
Discussion	96
More information	97
<b>12. Service Discovery.....</b>	<b>99</b>
Problem	99
Solution	100
Internal Service Discovery	101
Manual Service Discovery	104
Service Discovery from Outside the Cluster	107
Application Layer Service Discovery	111
Discussion	113
More Information	115
<b>13. Self Awareness.....</b>	<b>117</b>
Problem	117
Solution	117
Discussion	121
More Information	121

---

## Part III. Structural Patterns

<b>14. Init Container.....</b>	<b>125</b>
Problem	125
Solution	126
Discussion	130
More Information	130
<b>15. Sidecar.....</b>	<b>131</b>
Problem	131
Solution	132
Discussion	134
More Information	134

<b>16. Adapter.....</b>	<b>135</b>
Problem	135
Solution	135
Discussion	138
More Information	138
<b>17. Ambassador.....</b>	<b>139</b>
Problem	139
Solution	139
Discussion	141
More Information	142

---

## Part IV. Configuration Patterns

<b>18. EnvVar Configuration.....</b>	<b>145</b>
Problem	145
Solution	145
Discussion	148
More Information	149
<b>19. Configuration Resource.....</b>	<b>151</b>
Problem	151
Solution	151
Discussion	156
More Information	156
<b>20. Immutable Configuration.....</b>	<b>157</b>
Problem	157
Solution	157
Docker Volumes	158
Kubernetes Init Containers	159
OpenShift Templates	162
Discussion	163
More Information	164
<b>21. Configuration Template.....</b>	<b>165</b>
Problem	165
Solution	165
Discussion	170
More Information	171



---

## Part V. Advanced Patterns

<b>22. Controller</b> .....	<b>175</b>
Problem	175
Solution	176
Discussion	186
More Information	187
<b>23. Operator</b> .....	<b>189</b>
Problem	189
Solution	190
Custom Resource Definitions	190
Controller and Operator Classification	192
Operator Development and Deployment	195
Example	197
Discussion	201
More Information	202
<b>24. Elastic Scale</b> .....	<b>203</b>
Problem	203
Solution	204
Manual Horizontal Scaling	204
Horizontal Pod Autoscaling	205
Vertical Pod Autoscaling	210
Cluster Autoscaling	213
Scaling Levels	216
Discussion	219
More Information	219
<b>25. Image Builder</b> .....	<b>221</b>
Problem	221
Solution	222
OpenShift Build	223
Knative Build	230
Discussion	234
More Information	235
<b>Afterword</b> .....	<b>237</b>
<b>Index</b> .....	<b>239</b>

---

# Foreword

When Craig, Joe, and I started Kubernetes nearly five years ago, I think we all recognized its power to transform the way the world developed and delivered software. I don't think we knew, or even hoped to believe, how quickly this transformation would come. Kubernetes is now the foundation for the development of portable, reliable systems spanning the major public clouds, private clouds, and bare-metal environments. However, even as Kubernetes has become ubiquitous to the point where you can spin up a cluster in the cloud in less than five minutes, it is still far less obvious to determine where to go once you have created that cluster. It is fantastic that we have seen such significant strides forward in the operationalization of Kubernetes itself, but it is only a part of the solution. It is the foundation on which applications will be built, and it provides a large library of APIs and tools for building these applications, but it does little to provide the application architect or developer with any hints or guidance for how these various pieces can be combined into a complete, reliable system that satisfies their business needs and goals.

Although the necessary perspective and experience for what to do with your Kubernetes cluster can be achieved through past experience with similar systems, or via trial and error, this is expensive both in terms of time and the quality of systems delivered to our end users. When you are starting to deliver mission-critical services on top of a system like Kubernetes, learning your way via trial and error simply takes too much time and results in very real problems of downtime and disruption.

This then is why Bilgin and Roland's book is so valuable. *Kubernetes Patterns* enables you to learn from the previous experience that we have encoded into the APIs and tools that make up Kubernetes. Kubernetes is the by-product of the community's experience building and delivering many different, reliable distributed systems in a variety of different environments. Each object and capability added to Kubernetes represents a foundational tool that has been designed and purpose-built to solve a specific need for the software designer. This book explains how the concepts in Kubernetes solve real-world problems and how to adapt and use these concepts to build the system that you are working on today.

In developing Kubernetes, we always said that our North Star was making the development of distributed systems a CS 101 exercise. If we have managed to achieve that goal successfully, it is books like this one that are the textbooks for such a class. Bilgin and Roland have captured the essential tools of the Kubernetes developer and distilled them into segments that are easy to approach and consume. As you finish this book, you will become aware not just of the components available to you in Kubernetes, but also the “why” and “how” of building systems with those components.

— *Brendan Burns,*  
*Cofounder, Kubernetes*

---

# Preface

With the evolution of microservices and containers in recent years, the way we design, develop, and run software has changed significantly. Today’s applications are optimized for scalability, elasticity, failure, and speed of change. Driven by new principles, these modern architectures require a different set of patterns and practices. This book aims to help developers create cloud-native applications with Kubernetes as a runtime platform. First, let’s take a brief look at the two primary ingredients of this book: Kubernetes and design patterns.

## Kubernetes

*Kubernetes* is a container orchestration platform. The origin of Kubernetes lies somewhere in the Google data centers where Google’s internal container orchestration platform, **Borg**, was born. Google used Borg for many years to run its applications. In 2014, Google decided to transfer its experience with Borg into a new open source project called “Kubernetes” (Greek for “helmsman” or “pilot”), and in 2015, it became the first project donated to the newly founded Cloud Native Computing Foundation (CNCF).

Right from the start, Kubernetes gained a whole community of users, and the number of contributors grew at an incredibly fast pace. Today, Kubernetes is considered one of the most active projects on GitHub. It is probably fair to claim that at the time of this writing, Kubernetes is the most commonly used and feature-rich container orchestration platform. Kubernetes also forms the foundation of other platforms built on top of it. The most prominent of those Platform-as-a-Service systems is Red Hat OpenShift, which provides various additional capabilities to Kubernetes, including ways to build applications within the platform. These are only some of the reasons we chose Kubernetes as the reference platform for the cloud-native patterns in this book.

This book assumes you have some basic knowledge of Kubernetes. In **Chapter 1**, we recapitulate the core Kubernetes concepts and lay out the foundation for the following patterns.

# Design Patterns

The concept of *design patterns* dates back to the 1970s and from the field of architecture. Christopher Alexander, an architect and system theorist, and his team published the groundbreaking *A Pattern Language* (Oxford University Press) in 1977, which describes architectural patterns for creating towns, buildings, and other construction projects. Sometime later this idea was adopted by the newly formed software industry. The most famous book in this area is *Design Patterns—Elements of Reusable Object-Oriented Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides—the Gang of Four (Addison-Wesley). When we talk about the famous Singleton, Factories, or Delegation patterns, it’s because of this defining work. Many other great pattern books have been written since then for various fields with different levels of granularity, like *Enterprise Integration Patterns* by Gregor Hohpe and Bobby Woolf (Addison-Wesley) or *Patterns of Enterprise Application Architecture* by Martin Fowler (Addison-Wesley).

In short, a pattern describes a *repeatable solution to a problem*.<sup>1</sup> It is different from a recipe because instead of giving step-by-step instructions to solving a problem, a pattern provides a blueprint for solving a whole class of similar problems. For example, the Alexandrian pattern “Beer Hall” describes how public drinking halls should be constructed where “strangers and friends are drinking companions” and not “anchors of the lonely.” All halls built after this pattern look different, but share common characteristics such as open alcoves for groups of four to eight and a place where a hundred people can meet with beverages, music, and other activities.

However, a pattern does more than provide a solution. It is also about forming a language. The unique pattern names form a dense, noun-centric language in which each pattern carries a unique *name*. When this language is established, these names automatically evoke similar mental representations when people speak about these patterns. For example, when we talk about a table, anyone speaking English assumes we are talking about a piece of wood with four legs and a top on which you can put things. The same thing happens in software engineering when we talk about a “factory.” In an object-oriented programming language context, we immediately associate with a “factory” an object that produces other objects. Because we immediately know the solution behind the pattern, we can move on to tackle yet unsolved problems.

---

<sup>1</sup> Christopher Alexander and his team defined the original meaning in the context of architecture as follows: “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice,” (*A Pattern Language*, Christopher Alexander et al., 1977, p. x). We think this definition works for the patterns we describe in this book, except that we probably don’t have as much variability in our solutions.

## Kubernetes Patterns

The way developers design, build, and run software has changed significantly with the evolution of microservices and containers. These modern architectures offer new distributed primitives that require a different set of practices than many developers, tech leads, and architects are accustomed to. This focused guide provides common, reusable patterns and principles for designing and implementing cloud-native applications on Kubernetes.

Each pattern includes a description of the problem and a Kubernetes-specific solution. All patterns are backed by and demonstrated with concrete code examples. This book is ideal for developers and architects already familiar with basic Kubernetes concepts who want to learn how to solve common cloud native challenges with proven design patterns.

You'll learn about the following pattern categories:

- **Foundational patterns** cover core principles and practices for building container-based cloud-native applications
- **Behavioral patterns** explore finer-grained concepts for managing container and platform interactions
- **Structural patterns** help you organize containers within a Pod for addressing specific use cases
- **Configuration patterns** provide insight into how application configurations can be handled in Kubernetes
- **Advanced patterns** cover more complex topics such as operators and autoscaling

Bilgin Ibryam (@bibryam) is a principal architect at Red Hat and a committer to multiple projects at the Apache Software Foundation.

Roland Huß (@ro14nd) is a principal software engineer at Red Hat and a member of the serverless team working on Knative.

KUBERNETES / SOFTWARE DEVELOPMENT

For sale in the Indian Subcontinent (India, Pakistan, Bangladesh, Nepal, Sri Lanka, Bhutan, Maldives) and African Continent (excluding Morocco, Algeria, Tunisia, Libya, Egypt, and the Republic of South Africa) only. Illegal for sale outside of these countries.



MRP: ₹ 750.00

Twitter: @oreillymedia  
facebook.com/oreilly

**SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.**

"As you finish this book, you become aware of not just the components that are available to you in Kubernetes but also the why and how of building systems with those components."

—Brendan Burns  
Cofounder, Kubernetes

"A unique approach that introduces key Kubernetes concepts in a format that developers can understand and quickly take advantage of."

—Andrew Block  
Senior Principal Consultant, Red Hat

"A great book that explains Kubernetes aligned with real-world tasks and challenges."

—Michael Hüttermann  
Principal DevOps Consultant,  
Huettermann.net

ISBN : 978-93-5213-826-5



9 789352 138265

First Edition/2019/Paperback/English