Certified Kubernetes Application Developer (CKAD) Exam Success Guide

Ace your career in Kubernetes development with CKAD certification

Sethumadhavan Kishore

www.bpbonline.com

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55515-698

www.bpbonline.com

Dedicated to

My beloved Parents:

K P Sethumadhavan

Girija Sethumadhavan

&

My wife ***Ambili*** *and my little tiger* ***Aarav***

About the Author

Sethumadhavan Kishore is an esteemed IT professional with over a decade of experience in the industry. He has held various roles throughout his career, including software engineer, technical lead, and software architect. Kishore is well-versed in both Java and Python programming languages and has extensive experience working with distributed systems, Kubernetes, and OpenShift.

Kishore has been deeply involved in cutting-edge technologies throughout his career, and his expertise in Kubernetes is unparalleled. With a keen understanding of containerization and orchestration, he has successfully navigated complex deployments, leveraging Kubernetes to streamline processes, optimize resource utilization, and enhance scalability.

Kishore holds various certifications in Kubernetes. He is a Certified Kubernetes Application Developer (CKAD), Certified Kubernetes Administrator (CKA) and Certified Kubernetes Security (CKS). He scored 96% in the CKAD exam.

Kishore is an Innovator, and has successfully filed patents, published technical papers, and presented at regional technical exchanges and seminars. He has a proven track record of delivering high-quality software solutions and has a deep understanding of the intricacies of building and deploying large-scale systems.

When not immersed in technology, Kishore enjoys spending time with his family, exploring the outdoors, recording technical videos for his YouTube channel and continuously expanding his horizons by diving into new areas of interest within the IT landscape.

About the Reviewers

- **Keshav Saxena** is a seasoned DevOps engineer with over 8 years of experience in the IT industry. He has a strong background in cloud computing, having worked extensively with major cloud platforms such as AWS and GCP. His primary expertise lies in containerization and orchestration technologies, particularly Docker and Kubernetes. He has successfully implemented and managed containerized applications at scale, optimizing performance and efficiency. Keshav is proficient in infrastructure-as-code tools like Terraform and Ansible. He has leveraged these tools to automate the provisioning and configuration of cloud resources, enabling rapid deployment and easy scalability. Keshav's commitment to professional growth is evident through his certifications in Kubernetes, Terraform, and GitLab. These certifications validate his deep understanding and practical experience in these areas, making him a valuable asset to any organization. He loves reading books and watching documentaries.

- **Pratikkumar Panchal** has over eight years of professional experience. He is a skilled DevOps Cloud Engineer specializing in Kubernetes orchestration. Their expertise lies at the intersection of cloud infrastructure management and software development, enabling seamless deployment and continuous integration of complex applications. Throughout their career, Pratikkumar Panchal has demonstrated a deep understanding of cloud technologies, utilizing platforms like AWS, Azure, or Google Cloud to architect scalable and resilient solutions.

  As a Kubernetes engineer, Pratikkumar Panchal has a proven track record of designing, implementing, and maintaining Kubernetes clusters, streamlining application deployment and management. Their proficiency in containerization, combined with their ability to automate processes using tools like Docker and Jenkins, has resulted in optimized workflows and increased efficiency.

  Pratikkumar Panchal is dedicated to ensuring the reliability and security of systems, employing best practices in monitoring, logging, and infrastructure as code. Their strong collaboration skills have facilitated cross-functional teamwork, leading to successful project deliveries. Whether optimizing infrastructure, enhancing CI/CD pipelines, or troubleshooting complex issues, Pratikkumar Panchal continues to drive innovation and excellence in the DevOps and cloud engineering landscape.

Acknowledgements

I would like to express my deepest gratitude to the following individuals whose unwavering support and encouragement have made this book possible.

First and foremost, I am indebted to my loving family. Their patience, understanding, and belief in me have inspired me constantly. I am profoundly grateful to the dedicated team at BPB. Furthermore, I would like to acknowledge the invaluable contributions of the technical reviewer, Keshav.

I am also thankful for the support and encouragement received from my friends and colleagues. Your words of encouragement, stimulating discussions, and belief in my abilities have been instrumental in this undertaking.

Lastly, I want to extend my heartfelt appreciation to all the readers who will embark on this literary journey. Your curiosity and interest in the subject have fueled my writing passion.

In conclusion, completing this book has been a collaborative effort, and I am profoundly grateful to each and every individual who has contributed to its creation. This endeavor would not have been possible without your unwavering support, belief, and guidance.

Preface

Welcome to the Preface of this book, designed to assist you in your preparation for the Certified Kubernetes Application Developer (CKAD) exam. Whether you are an experienced developer or new to Kubernetes, this resource aims to provide you with the knowledge, skills, and confidence needed to excel in the CKAD certification process.

The CKAD exam comprehensively assesses your ability to design, develop, and deploy applications on Kubernetes clusters. It challenges your understanding of core concepts, such as creating and configuring application deployments, services, and persistent storage, as well as troubleshooting and debugging common issues within a Kubernetes environment.

Having personally navigated the CKAD certification journey, I understand the significance of a well-rounded and practical approach to exam preparation. This book results from my firsthand experience and extensive research and insights gathered from fellow CKAD-certified professionals. It has been meticulously crafted to equip you with the essential knowledge and hands-on skills necessary to ace the exam.

The chapters of this book are structured to follow the CKAD exam syllabus, offering a comprehensive and systematic approach to learning the required topics. Each chapter provides a clear explanation of the underlying concepts, accompanied by practical examples and step-by-step instructions to ensure a solid understanding of the subject matter. In addition, you will find exercises and practice scenarios throughout the book that simulate real-world situations you may encounter in the CKAD exam.

I would like to express my deepest gratitude to the team at BPB Publications, for their support and guidance in bringing this book to fruition. Their dedication and expertise have played a crucial role in ensuring the accuracy and relevance of the content.

Lastly, I extend my heartfelt thanks to all the readers who have chosen this book as a part of their CKAD exam preparation journey. Your commitment to expanding your knowledge and skills is commendable, and I am honored to be a part of your learning experience.

Best of luck in your CKAD exam preparation, and may this book serve as a valuable resource to help you achieve your certification goals.

The book has two simple objectives:

- Help you to master Kubernetes.
- Help you to crack the CKAD exam.

There is currently a high demand for Kubernetes skills in the job market. According to a survey by the Cloud Native Computing Foundation (CNCF), Kubernetes is the most in-demand skill for cloud-native professionals. Many companies are adopting Kubernetes to manage their containerized applications and infrastructure, and as a result, there is a growing need for professionals with expertise in using Kubernetes to design, deploy, and maintain cloud-native systems.

Thus, learning and mastering Kubernetes is essential.

This book is divided into **14 chapters**. The details are listed below:

Chapter 1: Introduction to Kubernetes, Docker, and Minikube - In this foundation chapter, we will learn how software development and deployment have changed over the years and also do a deep dive and try to understand the world of Cloud native technologies, Container, Docker and Kubernetes. We will also learn how to set up your Minikube. Minikube is a tool that allows you to run a single-node Kubernetes cluster locally on your own machine. Minikube single node cluster will be used to execute all the practice exercises.

Chapter 2: What, Why, and How of CKAD Exam - This chapter discusses the certification, curriculum and tips and tricks needed to pass the CKAD exam. We will also cover the topic of time management and how to practice and prepare for the CKAD exam.

Chapter 3: Exploring Pod, Deployment, ReplicaSet, and Namespace - From this chapter, we will start learning Kubernetes resource objects. First, we will start learning the core concepts of Kubernetes with Pod, Deployments, ReplicaSet and namespace. We will also discuss the basic structure of Kubernetes primitives and how to interact with Kubernetes objects via the command line–based client, kubectl. At the end of the chapter, you will understand how to create Kubernetes objects imperatively and declaratively and know how to create core Kubernetes objects, and define their most basic configuration.

Chapter 4: Deployment Strategies and Helm Packages - In this chapter, we will learn how to update apps running in a Kubernetes cluster and in an actual zero-downtime update process. We learn different deployment strategies and also learn Helm, the package manager, which is a tool to help you define, install, and upgrade applications running on Kubernetes.

Chapter 5: Pod Design and Concepts - In this chapter, you will learn many concepts related to Pod and deployment, which is essential to know to pass the exam. This chapter will teach us about labels, annotations, and deployment. We will also learn about autoscaler, how to roll out a new revision of your application, and how to roll back to a previous revision. And lastly, we'll touch on the Kubernetes primitives Job and CronJob.

Chapter 6: Multi-container Pods and Design Patterns - In this chapter, we will learn about Multi-container Pod, its need, use cases and design Patterns. We will also look into Init Container.

Chapter 7: Kubernetes Volumes and Persistence - In this chapter, we will cover Kubernetes volume types. The chapter covers, different types of volumes for reading and writing data. We will also learn how to create and configure them. Kubernetes models persist data with two primitive PV and PVC. In this chapter, we will cover Persistence volume and Persistence volume claim, how to set it, and its relationship with Pod. We will also cover the Storage class, how to set it, and its creation using PersistentVolume.

Chapter 8: Configuring Volume, Environment Variables, ConfigMap, and Secrets - In this chapter, we will learn some advanced concepts to configure the Kubernetes object. We will start first with how to configure runtime behavior using environment variables. How to configure apps through config maps and how to handle sensitive information in Kubernetes using secrets. We will also cover Volumes and how to mount them to Pod.

Chapter 9: Service Accounts Resource Quota and Security Contexts - In this chapter, we will learn how to define security context to define privilege and access control settings. We will also learn the configuration needed to assign a Service Account to a Pod and also learn about resource boundaries and how to set resource quota.

Chapter 10: Liveness and Readiness - Probes are used by Kubernetes as an indication of a pod's health and state. Health probes help automate detection and correction of issues seen in the Kubernetes cluster. These probes help in checking if the application is working as expected. In this chapter, we will discuss the various probes, namely readiness, liveness and startup probes, the difference between liveness and readiness probes and under which scenario to use. We will also learn how to configure them and how to debug scenarios associated with liveness and readiness probes.

Chapter 11: Monitoring and Debugging of Kubernetes Cluster - In this chapter, we will see how to monitor and debug Kubernetes clusters. You will learn strategies for debugging a misconfigured or misbehaving Kubernetes object. Also, we will see how to monitor clusters and nodes.

Chapter 12: Kubernetes Networking and Services - This chapter will focus on various types of Kubernetes services and will also learn about exposure of Pods inside or outside of the cluster, depending on their declared types. You will also learn Kubernetes networking and network policy.

Chapter 13: Mock Exam 1 - This chapter will cover the first mock exams with solutions. The questions in these exams will give you a real exam-like feel regarding your ability to read and interpret a given question, how to manage time, and validate and debug your own work.

Chapter 14: Mock Exam 2 - This chapter will cover the second mock exam with solutions. The questions in these exams will give you a real exam-like feel regarding your ability to read and interpret a given question, how to manage time, and validate and debug your own work.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/p199ivx

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Certified-Kubernetes-Application-Developer-CKAD-Exam-Success-Guide**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to Kubernetes, Docker, and Minikube

Introduction

In this foundation chapter, we will learn how software development and deployment have changed over the years and dive deep into understanding the world of Container, Docker, and Kubernetes.

We will also learn about how to set up your minikube. minikube is a tool that allows you to run a single-node Kubernetes cluster locally on your own machine. minikube single node cluster will be used to execute all the practice exercises.

Structure

In this chapter, we will cover the following topics:

- Software development, deployment, and evolution
- Evolution of microservice
- Introducing Docker
- Introducing Kubernetes
- Architecture of Kubernetes
- Benefits of Kubernetes

- Installing Docker
- Installing minikube
- Practice test

Objectives

By the end of this chapter, the reader can understand how software development has changed over the years, the role of containers and virtual machine in the evolution of microservice, Kubernetes architecture. I hope this chapter provides a solid foundation.

Software development, deployment, and evolution

Around 15 years back, if our team wanted to deploy an application or run a service, we had to provision a whole new physical server. Back then, if we were developing an application such as a J2EE application, the team needed to work with the purchasing department to order physical hardware. It would take 4-8 weeks for the hardware to arrive, then the IT team would install OS with the required binaries/software updates and hand over the server to the project team. Only then the project team could deploy the application and initiate user testing before roll-out to production. So, from project inception to actual production, it used to take anywhere between 4-6 months. It was a frustrating, expensive, non-efficient, and time-consuming process. The scaling was costly and difficult, as it entitled buying more hardware.

Cut short to today; we can provision and spin up a new server and deploy applications within a few hours. All the provisioning activities can be conducted using a self-help portal, and there is no dependency on the IT team. Over time, the industry has evolved with technologies such as virtualization and containers. Let us try to understand this evolution from a physical server to virtualization to containers with the help of examples.

Deploying applications on a physical server

Resources are needed for a service or application to run on a physical server. The resources are **Memory**, **Central Processing Unit** (**CPU**), **Input/Output** (**I/O**), and **disk space**. These resources are managed by the operating system, as shown in the following figure: there is a hardware layer of the physical server with resources. On the hardware, the operating system is loaded. The application runs on the OS, which helps manage the resource utilization for the application.

In *Figure 1.1*, we can see how the resources are managed by the operating system:

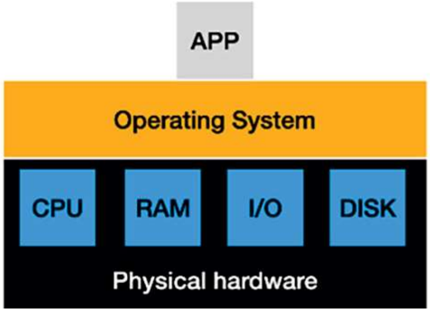

Figure 1.1: *Physical* server

Typically, the physical server architecture is single-tenant; that is, you only have one application running on a physical server. If multiple applications are running on a physical server, one application may consume most of the compute resources and impact other applications running on the same server. This is a common problem of shared resources without strong boundaries. Hence, a physical server runs only one application in most cases, thus not efficiently utilizing the physical server resources.

Moreover, most of the time, nobody knew the performance requirements of the new application. Procuring a new server would take several months, forcing the IT department to make guesses while choosing the model and size of the server to buy.

Thus, IT department used to buy big fast servers to run the applications, and in most instances, these servers were underutilized. It was a total waste of capital resources and money.

Deploying applications in Virtual Machines

Sometime around the mid-2000s, **Virtual Machines (VMs)** came into existence, where each server with VM technology could run multiple applications. The virtual machine was a game changer. With the advent of technologies such as VMware and hypervisor, organizations can utilize physical servers more efficiently. They can now allocate, add, and remove resources must faster. So, there was a drastic improvement in how servers were procured for applications, as new servers were no longer procured for every new application. With VM technologies, multiple applications can reside on a single server application, and the same physical server can be shared across multiple applications.

With a virtual machine, you have the same physical server with all the resources, but instead of the server operating system, there is a hypervisor such as vSphere or Hyper-V loaded to it. The Hypervisor is where you create your virtual machine.

In *Figure 1.2*, each VM has its own virtual device, such as a virtual CPU and virtual RAM. On top of these virtual hardware, you load a guest operating system and the applications. Refer to the following figure:

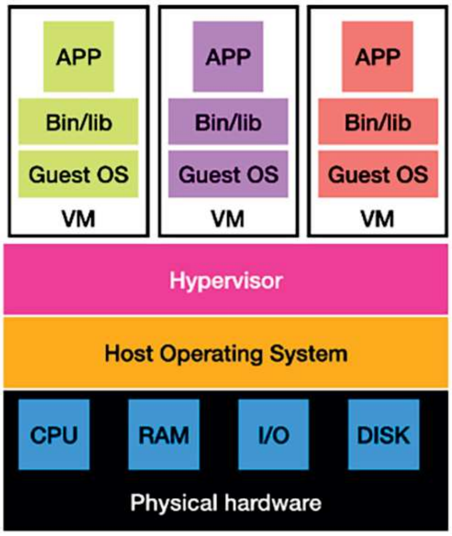

Figure 1.2: *Virtual machine*

Application services are deployed to each of these virtual machines, so there is a tight boundary and no sharing of the resources. The benefits of virtualization are obvious; instead of having just one application per server, you can now run several guest operating systems and a handful of applications with the same physical hardware.

Deploying applications in containers

So how are containers different from VMs? In short, containers virtualize the operating system instead of the hardware. Containers offer enhanced portability and efficiency by bundling code and dependencies together. Functioning as an abstraction at the application layer, they provide a seamless way to package and run software applications in any environment.

Multiple containers can run on the same machine and share the operating system kernel with other containers. Each of these containers runs as isolated processes. Compared to VMs, containers are only megabytes in size and take just seconds to start, versus gigabytes and minutes for a VM.

In the following figure, containers sit on top of a physical server and its host operating system (either Linux or Windows):

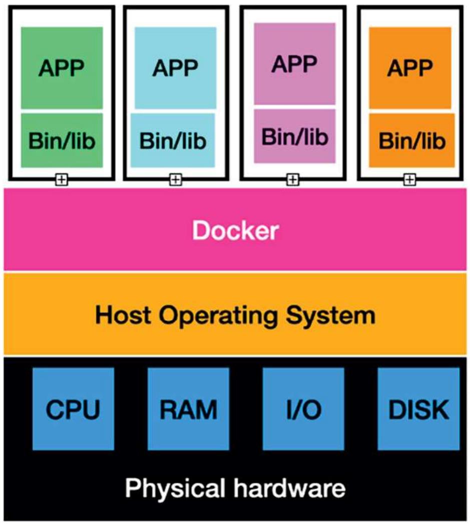

Figure 1.3: *Container*

Docker is a tool that uses containers to make the creation, deployment, and running of applications a lot easier. It binds the application and its dependencies inside a container.

Evolution of microservices

In the previous sections, we looked at how the software industry has evolved from physical servers to virtual machines to containers. This evolution has also impacted the software architecture, its deployment, and its development.

During the era of physical servers, most of the applications were built as monolithic applications. In monolithic applications, all components must be developed, deployed, and managed as one entity. Any change in one of the application's components required the redeployment of the whole application.

To handle the increase in load on a monolithic system, you could either vertically scale the system by adding more CPUs and memory or other server components. You can also scale horizontally, that is, by setting up additional servers and running multiple copies of the application. Scaling out horizontally has its own challenges, and you will need to re-architect the application if you want to scale out certain components of the application. The components of the applications are tightly coupled to the scale-out. Most of the time, you end up scaling out the whole components of the application, even though the intent was to scale only certain components. So, the solution was to split up this complex monolithic application into independently