Mastering DevOps in Kubernetes

Maximize your container workload efficiency with DevOps practices in Kubernetes

Soumiyajit Das Chowdhury

www.bpbonline.com

First published: 2024

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

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-309

www.bpbonline.com

Dedicated to

My beloved Parents:

Mr. Ramkrishna Das Chowdhury

Mrs. Shova Das Chowdhury

&

*My wife **Gitali** and my son **Vivaan***

About the Author

Soumiyajit Das Chowdhury, has a B.Tech in Electronics and Instrumentation, and a total of 17 years of experience in IT Industry. Soumiyajit has always been passionate about learning new software languages. He has worked on projects that involve software languages such as C, C++, Java, Python and Golang. He has also worked on multiple automation projects involving terraform and ansible. Soumiyajit has worked in the domains of telecom and networking. He has been working in the Cloud and Container domain for the past 11 years, and as an Architect/SME for AWS, Azure, GCP, OpenStack, Kubernetes, OpenShift and DevOps practice for the past 7 years. Soumiyajit has been a part of the OpenSouce community, such as OpenStack and CNCF, for a long time and has published multiple whitepapers in his past roles.

Acknowledgements

First and foremost, I would like to thank my parents for encouraging me to write this book. They have always taught me to keep learning, in a world of quick changing technologies. I would also like to thank all the teams I have worked with, in the past. My work experience has contributed a lot to my technical skills and knowledge.

In the recent past, I had the privilege to work with some of the best architects in the Cloud and DevOps industry, from whom I have learned profoundly, broadened my skills, and strengthened my craft and passion. Special thanks to my fellow architect and mentor Mr. Charles Nicholson. I would also like to thank Mr. Koji Shiotani for guiding me all these years and finding an architect in me.

My gratitude also goes to the entire team at BPB Publications for being supportive enough to provide me valuable feedbacks and guiding me as I drafted the book. They have provided me with enough time to come up with an initial draft of the book. BPB has a strong team of editors who have helped me throughout the process, and I really enjoyed writing with them.

Preface

Kubernetes is a fast-evolving container orchestrator, and alongside Kubernetes, we have a lot of tools and technologies that cater different use cases required in a real world. Kubernetes has a very strong use case of cost. Hence, the industry is moving quickly to container platform. However, there are certain intricacies. A lot of applications are large complex monoliths, and a few of them also have sticky sessions, a few of them are hard to adhere by the micro-service design, and so on. With all these complexities, organizations struggle to find the best DevOps practice required for their use cases.

This book is focused on answering the questions that we come across as we decide to migrate our applications to Kubernetes. It is focused on learning the best practice for DevOps. The readers would acquire the required knowledge of the DevOps practice before taking a decision about what works best for them. This book is also good for the candidates who have a basic knowledge of Kubernetes and would like to enhance their knowledge on standard tools and methodology that are adopted for applications to be managed and hosted in Kubernetes. A lot of organizations/teams use public cloud such as AWS, Azure or GCP. In this book, we have dedicated sections for the top three public cloud flavors of Kubernetes such as Amazon EKS, Azure AKS and GKE.

This book takes a practical approach for DevOps practice in Kubernetes. All the codes used in the book are present in the GitHub repository. This book is divided into 13 chapters. We begin with the problems in the DevOps practice and then deep dive into each chapter and learn how Kubernetes helps us solve these issues. The details of the chapters are listed as follows:

Chapter 1: DevOps for Kubernetes – will cover the issues faced by the Enterprise DevOps in current times. It also explains the traits and capabilities that come with Kubernetes, that make it useful for building, deploying, and scaling enterprise grades DevOps managed applications.

Chapter 2: Container Management with Docker – will cover Docker in depth including the Dockerfile, docker instructions and creating docker images from Golang and Python Flask Applications. It also explains about creating multi container images using docker compose. Readers would learn about efficient ways of managing containers using Docker and explore Docker swarm.

Chapter 3: Speeding up with Standard Kubernetes Operations – will cover Kubernetes Architecture in depth. We will also learn to create our own development clusters in the form of Minikube and Stand-Alone Kubernetes Cluster. We will also learn about the Standard Kubernetes Operations like Deployments, ConfigMaps, Autoscaling, Affinity, Jobs, etc. All the sections have Hands on for better understanding of the readers.

Chapter 4: Stateful Workloads in Kubernetes – will cover the concepts for Stateful Applications. The readers would be able to know how Persistent Volume and Persistent Volume Claim functions to manage the Kubernetes storage. This chapter also explains about dynamic storage provisioning based on the API object StorageClass. Readers would also learn how to create Stateful Applications in Kubernetes and the best practices required to implement StatefulSets in Kubernetes.

Chapter 5: Amazon Elastic Kubernetes Service – will cover AWS EKS in depth. Readers will know how to provision the AWS EKS Cluster. They would also learn how to provision the Amazon Elastic Block Storage to use in our AWS EKS Cluster. It also explains the use of managed databases in AWS EKS cluster using the AWS RDS service. Readers also learn to provision the different kind of load balancers like Classic, Network and Application Load Balancers.

Chapter 6: Azure Kubernetes Service – will cover Azure AKS in depth. Readers would be able to learn some of the important features and capabilities of Azure AKS. It explains how to manage AKS cluster through CLI and talks about the Azure Virtual Networks. It also explores the AKS Storage Class and Provisioners. Readers would also create Virtual Node for the existing cluster using the Azure Container Interface (ACI) to understand the serverless features in AKS.

Chapter 7: Google Kubernetes Engine – will cover GKE in depth. Readers would be able to know some of the key features and capabilities of Google Kubernetes Engine (GKE). They would learn to Provision a GKE cluster, and explore the options in GKE to Load Balance the service traffic beyond Cluster IP or Node Port. The chapter also explains how to configure a Service Type Load Balancer and also configure Load Balancing with Ingress Object. Readers would also verify how the cluster autoscaler works and learn about the Storage Provisioning in GKE and also access Cloud SQL from GKE application.

Chapter 8: Kubernetes Administrator – will cover some of the advanced features of Kubernetes. Readers would be able to learn about the use of resource quota, Networking and Network Policies in Kubernetes. It explains about the Node

Maintenance, Pod Disruption Budget and Pod Topology Spread Constraints. Readers would learn about the best practice to achieve the High Availability in Kubernetes. They would also learn about taking backup in etcd, and explore Kubernetes Probes such as Startup, Liveness and Readiness Probes.

Chapter 9: Kubernetes Security – will cover Kubernetes security. Readers would explore about Node Restrictions and how it can be useful to secure our nodes. This chapter also explains the use of static analysis tools like kubesec to analyze manifests in the development phase. It also talks about the use of security context and how we can restrict the access to our container and Pods in our manifest file. Readers would explore the use of Pod Security Admission and how policies can be used to secure Pods using different Pod admission modes.

Chapter 10: Monitoring in Kubernetes – will cover Monitoring using Prometheus and Grafana. Readers would learn with a sample application to use the Prometheus Client Library to scrape different kind of metrics to Prometheus and Grafana. This chapter also explores the AWS AKS Monitoring using the CloudWatch, AKS Monitoring and GKE Monitoring Stack.

Chapter 11: Packaging and Deploying in Kubernetes – will cover the different capabilities of Helm as a Kubernetes Package Manager. Readers would learn to deploy applications using Helm Charts. It also provides guidelines to create own charts and repositories using Helm. Readers would also learn about the Helm Hooks and how they can be beneficial to execute task at different stages of a release lifecycle.

Chapter 12: Continuous Development and Continuous Deployment – will cover Skaffold and Flux CD. Readers would be able to learn about Skaffold and how we can use it for DevOps practice. It also explains how to install Skaffold and use it to synchronize the code changes to the cluster without making the effort to build, push or deploy the application manually. This chapter also explains about Flux CD which is a very useful tool to automatically deploy applications into the cluster and update the desired state based on the code changes in the repository.

Chapter 13: Managing Microservices Using Istio Service Mesh – will cover Istio Service Mesh. Readers will learn how to install the Istio Service Mesh and explore the capabilities of Istio Service Mesh such as Controlling traffic, Weight Based Routing, Security and Observability.

Code Bundle and Coloured Images

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

https://rebrand.ly/h36r13x

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Mastering-DevOps-in-Kubernetes**. 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.

Table of Contents

CHAPTER 1
DevOps for Kubernetes

Introduction

As organizations adopted DevOps, development and operations teams worked together to build pipelines and integrate multiple tools. Even though these tools work well together, the specialization required by each tool results in the toolchain becoming difficult to manage. Every time an individual component requires replacement or updates, the entire pipeline must be redeveloped for the new component to work well within the toolchain. Soon, DevOps found the solution to this problem in the form of containerization. By creating a modular infrastructure based on microservices that could run in containers, organizations created portable pipelines that were built on containers. This helped the DevOps engineers to add or modify tools without disrupting the whole process. However, as the DevOps teams moved to containerization, the problem of orchestration and scalability emerged.

This is where Kubernetes came in. Kubernetes enhances the quality of the DevOps process because of its capabilities, such as consistency in development and deployment, compatibility with multiple frameworks, effortless scalability, self-healing capability, and many more.

Let us try to understand the challenges for DevOps in the real world and see how Kubernetes can help us mitigate those challenges.

Structure

The topics that will be covered in this chapter are as follows:

- Challenges for the enterprise DevOps
    - Managing multiple environments
    - Scalability and high availability
    - Implementation cost
    - Traffic management
    - Application upgrades and rollbacks
    - Securing infrastructure
    - Optimization of the delivery pipeline
    - Choice of tools and technology adoption
- Kubernetes DevOps
    - Infrastructure and configuration as code
    - Upgrading infrastructure
    - Updates and rollback
    - On-demand infrastructure
    - Reliability
    - Zero downtime deployments
    - Service mesh
    - Infrastructure security

Objectives

By the end of this chapter, the reader will be able to understand the issues faced by Enterprise DevOps in current times. We will also learn about many traits and capabilities that come with Kubernetes that make it useful for building, deploying, and scaling enterprise grades DevOps-managed applications.

Challenges for the enterprise DevOps

Before the DevOps days, the development and operations teams operated in silos. Each team had independent processes, goals, and tooling. These differences often created conflict between the teams and led to bottlenecks and inefficiencies. With the adoption of DevOps, many of these issues were resolved. DevOps resolves this by requiring cultural changes within the development and operations teams, which forces processes and workflows to overlap and run in tandem. However, these cultural changes were not enough to overcome all the issues that exist with siloed teams. Some of the challenges faced by Enterprise DevOps are addressed as follows.

Managing multiple environments

In the process of DevOps adoption, most of the organizations have sorted out some procedures for **Continuous Integration** and **Continuous Deployment** (**CI/CD**), and they manage codes to deploy and verify in each stage, all the way to production. To ensure this, teams require multiple environments where developers can deploy the application and confirm that the code is bug-free. Beyond Development, we also need to manage multiple environments, predominantly for Staging and Production. With a well-tuned workflow, the teams are not only productive, but it also helps them deliver software in a more reliable and timely manner. Some of the major advantages of using multiple environments are as follows:

- Using multiple environments in production reduces downtime, and thus, saves the organization from loss of revenue.
- Managing multiple environments also provides better security. Each team is provided with precise roles based on the environment. For example, a developer might not require access to production environments and, in certain cases, would only require view access. This helps the development teams from accidentally deleting production data. Similarly, there are numerous use cases for the kind of roles each should be provided, considering the principle of least privileges.
- Due to multiple environments in engineering teams, codes are verified multiple times to confirm that they are working as expected before moving to production. Moreover, the code gets tested with a variety of hardware and software configurations.
- Since the code is being managed across different branches, it is being verified in parallel across development and QA. This helps the product to move to production faster.

However, due to multiple changes in the environments related to infrastructure and configuration, most of the environments are not consistent.

Scalability and high availability

One of the measuring criteria for the success of an application is its ability to scale and make sure that the application is always available to the end users. In DevOps practice, the CI/CD process and synchronization of every moving object in the pipeline have minimized a few scalability issues. However, in most cases, the application still fails to have scalability on time because of one or more reasons, as discussed further:

- **Infrastructure and configuration:** In many cases, we use user-provisioned infrastructure, which triggers some script or code to provision new servers/ **Virtual Machines** (**VMs**) in case of vertical scaling. Moreover, in some cases, we use managed infrastructure to host our applications with no autoscaling enabled. In such cases, the frontend applications, at times, get too many requests, and by the time the scaling is achieved, there are quite a few users whose requests have already failed. All we need is a just-in-time scaling and on-demand infrastructure.
- **Scale cube of microservices:** In the case of a monolith, we can scale the application independently. However, in the case of a microservice application, we tend to scale just the component or service that we need to upscale. We should first find out the scaleup dependency and scale each application one by one. Only running multiple copies of an application behind the load balancer caches unnecessary memory and does not tackle the problem of application complexity. In multiple cases, the dependency of the application does not get reflected to scale in the *Y*-axis or *Z*-axis, as per the principle of scale cube of microservices. Either it is too much of a complex decomposition for the development team, or it is an architectural drawback.
- **Application delivery controller:** Application Delivery Controller is used by a lot of DevOps Teams that are having performance and efficiency issues. Some of them are limited to run on a single platform in a single location. In certain cases, they are not even compatible to work with a hybrid (servers and containers) stack of applications.
- **Operability distracts scalability:** As we design our systems to be more scalable, it becomes difficult for humans to operate them. The trade-off between operability and scalability may involve a loss of fine-grained human control of the system to achieve levels of scalability that would otherwise be unmanageable.
- **Logging and tracing tools:** Logging and tracing is a cross-cutting concern for DevOps engineers. It is imperative, in current times, to use logging for every application we use in production to trace and analyze errors, warnings, and