DevOps Automation Cookbook

Harness the power of DevOps with 125+ automation recipes

Ekambar Kumar Singirikonda

www.bpbonline.com

First Edition 2024

Copyright © BPB Publications, India

ISBN: 978-93-55519-061

Dedicated to

I would like to thank my family for their love and encouragement, who have been the rock of my journey. Gratitude is also owed to my coworkers and mentors, who have inspired me to strive for greatness in the realm of DevOps. A heartfelt appreciation goes out to the DevOps community for their knowledge and enthusiasm that fuels progress and shapes our industry.

About the Author

Ekambar Kumar Singirikonda (**Kumar**), serves as the Director of DevOps Engineering at Toyota North America. He leads performing teams and drives transformative initiatives within the organization.

Kumar specializes in DevOps engineering, cloud engineering, edge compute engineering and system performance analysis. Throughout his journey, he has introduced automation solutions that have significantly impacted businesses.

Kumar has received accolades such, as the Inspirational DevOps Leadership Team Award and the Quality Excellence Award. His writings encompass topics in the realm of DevOps, such, as customer experience, compliance, governance, security, data privacy and incident response.

In addition to his position at Toyota, Kumar is a board member at The University of Texas at Austin's McCombs School of Business. He also sits on the Board of Directors for Gift Of Adoption Funds, an organization that provides grants for child adoption.

As part of CDO Magazine's Editorial Board, Kumar discusses advancements in data and analytics. He is also involved with the council for the Harvard Business Review.

Living in Irving, Texas, with his family, Kumar actively participates in the DevOps community. He mentors aspiring professionals to help them improve their skills and expertise, in the field.

About the Reviewer

Pradeep S. Chintale is a seasoned lead cloud solutions architect with extensive experience in system analysis, infrastructure automation, and cloud/DevOps engineering. Currently based in Pennsylvania, USA, at SEI Investment Company, he excels in developing and designing robust DevOps architecture solutions, implementing scalable and secure cloud-based systems, and spearheading Kubernetes-based container orchestration.

With a bachelor's degree in computer science from Mumbai University, Pradeep has amassed over 18 years of experience in the field. His previous roles include serving as a Senior Cloud Solution Architect at Microsoft and a Lead Cloud/DevOps Engineer at Comcast, where he significantly contributed to developing innovative cloud solutions and enhancing operational efficiencies.

Pradeep is the lead author of the book "DevOps Design Patterns" published by BPB.

He has been a driving force in the tech community, serving as an industry judge at the Globee Awards in Cybersecurity and participating as a board member for Education and Innovation at The New World Foundation nonprofit organization.

His certifications and recognitions as a Cloud Solution Professional Architect and a top professional in DevOps and cloud technologies underscore his expertise and contributions to the field. Pradeep is a member of prestigious organizations such as IEEE and The Linux Foundation, reflecting his commitment to continual learning and professional development in the rapidly evolving tech landscape.

Acknowledgement

A special recognition to everyone who contributed to making the book, **DevOps Automation Cookbook**, a reality. Without the help, guidance, and expertise of individuals, this project would not have come together as it has. I am especially thankful to my colleagues at Toyota North America for their influence on my understanding of DevOps practices and their valuable insights during the writing process. Their input was crucial in refining the content for application in real world scenarios.

My gratitude extends to the publisher for their assistance in bringing this book to life. Their skill and dedication were instrumental in navigating through the complexities of publishing. To all reviewers, experts and editors whose feedback enhanced this manuscript, I am truly appreciative. Your attention to detail and constructive suggestions have greatly improved the quality and clarity of this work. I am grateful for all the reader interest in and support of this book.

I truly believe that this book will serve as a resource, for your exploration into the realm of DevOps automation.

Preface

In todays fast-paced world having an understanding of DevOps principles and automation is crucial for professionals across different industries. The **DevOps Automation Cookbook** aims to be a guide that covers the concepts, tools, and practices driving efficient software development and operations.

This book dives into topics encompassing DevOps principles and advanced automation strategies. Each chapter focuses on an aspect of DevOps automation, providing insights and practical tips to help readers implement these concepts in their projects.

Chapter 1: Introduction - This chapter offers an overview of the concepts of DevOps, highlighting the importance of automation in the DevOps workflow. It explores the history and evolution of DevOps, key principles, and the pivotal role played by automation. The chapter introduces tools and technologies that facilitate DevOps automation, such as Git, Jenkins, Docker, Kubernetes, Ansible, and cloud platforms.

The book presents guidelines for understanding the DevOps lifecycle, assessing an organizations readiness for DevOps adoption and establishing a DevOps pipeline. Key takeaways include discussing how automation in DevOps enhances speed, accuracy and scalability; along with suggestions on selecting tools tailored to project needs.

Chapter 2: Understanding Infrastructure as Code - This chapter explores the concept of Infrastructure as Code (IaC) and its role in managing and setting up infrastructure. It covers the transition from infrastructure management to IaC outlining the benefits, challenges and strategies to overcome them. The chapter also compares tools like Terraform, Ansible, Chef, Puppet, Pulumi and SaltStack. Practical examples illustrate the implementation of IaC using Terraform, along with testing using rollback techniques with Chef. Key takeaways include understanding core principles of methods, the advantages of adopting practices and choosing the right tools for IaC.

Chapter 3: Provisioning with Terraform - This chapter provides an overview of Terraform for infrastructure provisioning, highlighting its cloud agnostic nature and support for immutable infrastructure. It discusses Terraforms significance in DevOps and challenges when integrating systems not managed by Terraform. The chapter offers tips on maintaining configurations (DRY), handling state files cautiously and meticulously planning and reviewing changes. Examples showcase how Terraform can manage AWS resources such as EC2 instances S3 buckets, RDS databases, load balancers, and VPCs among others.

Essential insights focus on mastering Terraform, for infrastructure management and automating deployment procedures.

Chapter 4: Version Control with Git - In this chapter we take a dive into Gits version control capabilities, examining its distributed structure and the benefits it brings in terms of speed and efficiency. We explore aspects of Git such as commits, branches, merges, pull requests and how it fosters work on a larger scale. The practical guides walk you through the process of installing Git setting up repositories creating branches, merging workflows resolving conflicts efficiently and making use of features like tags and stashing. The main points to remember focus on understanding the principles of Git, becoming proficient in its development tools and incorporating Git into DevOps environments to systematically track code changes.

Chapter 5: Introduction to Continuous Integration with Jenkins and Travis - This chapter underscores the importance of **Continuous Integration (CI)** in integrating code changes and validating them promptly. It provides a comparison between two CI tools—Jenkins and Travis CI—discussing their backgrounds, features, strengths and recommended usage scenarios. The step by step guides offer instructions on setting up and managing build pipelines in Jenkins, configuring units, and integration tests, effectively deploying code securely, while establishing Travis CI for GitHub projects. They cover concepts such as mastering the basics of CI practices, distinguishing between Jenkins and Travis CI handling build failures, seamless integration of CI practices into DevOps, quick feedback loops, and deployable code.

Chapter 6: Automated Testing in DevOps - This chapter explores the role of automated testing in achieving integration and delivery within the DevOps framework. The chapter discusses types of testing such as unit testing, integration testing and end to end testing, highlighting their importance. Practical examples are provided to demonstrate the use of testing frameworks like unit testing frameworks for code coverage analysis using Istanbul, conducting UI tests with Selenium, implementing **behavior driven development (BDD)** with Cucumber and performing performance tests with JMeter, among other methods. Valuable insights are shared on understanding the significance of test automation and seamlessly integrating types of testing practices into DevOps workflows.

Chapter 7: Test Automation with Selenium - This chapter delves into exploring Seleniums capabilities in automating web browsers for test automation purposes. An overview is provided of Seleniums components, including Selenium IDE, WebDriver and Grid, along with their applications. The chapter details setting up Selenium configurations, crafting and executing test scripts efficiently, integrating them into integration pipelines, smoothly managing web elements effectively, while also covering tests using Appium.

The primary objectives revolve around comprehending the importance of Selenium in automating tests, creating test scripts, integrating Selenium into DevOps processes effectively and implementing strategies for test automation.

Chapter 8: Understanding Containers and Orchestration - This chapter discusses containerization and its benefits, in running applications across computing environments. It explains the differences between VMs and containers and clarifies Docker concepts such as images and containers. The chapter highlights Kubernetes role in managing containers its components and advantages in automating container operations. Practical examples demonstrate processes like Docker installation, image and container creation Kubernetes configuration, and application deployment, among others. Key points include grasping container basics and orchestration fundamentals, utilizing Docker and Kubernetes for deploying and managing applications.

Chapter 9: Deployment with Docker and Kubernetes - This chapter takes you through the process of deploying applications using industry-standard containerization technologies. This chapter explores the evolution of deployment practices from traditional to DevOps approaches. You will understand the essential components of Dockerfile, learn about Kubernetes manifests and their role in deployment, gain knowledge on container lifecycle management, understand the concept of pod orchestration, and learn troubleshooting techniques for deployment scenarios.

Chapter 10: Introduction to Security in DevOps - This chapter emphasizes the significance of integrating security practices into DevOps methodologies (DevSecOps) to ensure software integrity and compliance adherence. It discusses security challenges in DevOps, like vulnerabilities from insecure dependencies and misconfigurations, along with the importance of security tools. Practical guidance provides insights on security audits, establishing security monitoring using SIEM tools conducting security assessments with OWASP ZAP toolkit, among other topics. Key insights focus on understanding DevSecOps principles, implementing security measures, and integrating security seamlessly into DevOps workflows.

Chapter 11: Puppet and Security - This chapter delves into the role of Puppet in maintaining configurations and compliance, within DevOps settings. It explores Puppets structure, components and function in enforcing desired system states.

The examples demonstrate how Puppet can be utilized to enforce security policies, audit system configurations, address vulnerabilities, monitor systems and more. Key points to note include understanding Puppets security and compliance features, implementing security measures using Puppet and integrating Puppet into DevOps workflows.

Chapter 12: Configuration Management with Chef - The focus is on exploring Chef for configuration management. It discusses Chefs architecture, components and Ruby DSL while highlighting its effectiveness in managing large scale systems, supporting platforms and transforming infrastructure into code. Step by step guides in this chapter show how to set up Chef create cookbooks and recipes; utilize Chef for enhancing security measures and ensuring compliance. Important concepts covered include grasping the core principles of Chef, executing configuration management with Chef tools and incorporating Chef into DevOps methodologies.

Chapter 13: Ensuring Compliance with TeamCity - This chapter sheds light on TeamCitys role in facilitating Continuous Integration (CI) and **Continuous Delivery (CD)** processes while ensuring compliance standards within DevOps practices are met. It details TeamCitys functionalities related to managing builds storing artifacts gathering data from build tools for reporting purposes. The recipes, in the chapter demonstrate how to set up TeamCity configure build pipelines implement compliance policies troubleshoot build failures and more.

The main points highlight the importance of grasping TeamCity's functionalities in integration and continuous deployment (CI/CD) and compliance by establishing build processes and ensuring adherence, to TeamCity standards.

Chapter 14: Implications and Future Directions - This chapter concludes the book by summarizing the discussed concepts and tools from chapters. It underscores the importance of practices like IaC, CI/CD, automated testing, and configuration management in DevOps implementations. The chapter provides recommendations for learning opportunities through certifications, online resources, books and hands on projects. Additionally, it delves into DevOps trends such as artificial intelligence/machine learning (AI/ML) embracing serverless architectures, enhancing security measures, and exploring edge computing technologies, among others. Readers are encouraged to remain curious and adaptable as DevOps methodologies progress.

Code Bundle and Coloured Images

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

https://rebrand.ly/yyp8kfe

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/DevOps-Automation-Cookbook.
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.

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

Introduction

This chapter will help you understand the fundamental principles of DevOps and the importance of the DevOps lifecycle from coding to monitoring. You will gain an in-depth comprehension of the critical role automation plays in DevOps, including its impact on speed, accuracy, repeatability, and scalability of software development and operations. The chapter will increase your familiarity with the range of tools and technologies that facilitate DevOps automation, including Git, Jenkins, Docker, Kubernetes, Ansible, and cloud platforms.

DevOps is not something that just popped up overnight. It has been a journey of learning and adapting over time. Back in the early 2000s, there was a problem: development teams and operations teams did not always work well together. They were like two separate islands, causing delays and issues in getting software out to users.

Then, in 2009, something big happened. At a conference, folks from Flickr showed how they were doing something amazing: Deploying their software up to 10 times daily! This blew people's minds. It showed that by working together and using automation, teams could move faster and deliver better software.

From there, DevOps started to take off. In 2010, the first DevOpsDays conference happened in Belgium. It brought together people from both sides – developers and operations folks – to talk about how they could work better together.

Over time, DevOps has grown and changed. Automation has become a big part of it. Tools like Git, Jenkins, Docker, Kubernetes, Ansible, and cloud platforms have made it easier for teams to automate tasks, making things faster, more accurate, and scalable.

In this chapter, we will explore how automation makes everything tick. You will learn why it is so important and learn about some of the tools that make it all possible. By the end, you will be ready to take on DevOps with confidence!

Structure

This chapter will cover the following topics:

- The advent of DevOps and its importance
- Fundamentals and principles of DevOps
- Role of automation in DevOps
- Tools and technologies for DevOps automation
- Recipe 1: Understanding the DevOps lifecycle
- Recipe 2: Evaluating organizational readiness for DevOps adoption
- Recipe 3: Establishing a basic DevOps pipeline
- Key learnings
- Application in real-world scenarios

Objectives

By the end of this chapter, you will learn how to evaluate an organization's readiness for DevOps adoption, considering key influencing factors such as *culture*, *tools*, *workflows*, and communication styles. You will also acquire practical knowledge on establishing a basic DevOps pipeline, from setting up a Git repository to monitoring application performance after deployment.

Readers will develop an understanding of how to choose the right tools for their specific needs, considering factors like the nature of projects, team skills, and business requirements. You will be able to identify potential improvements and inefficiencies in their existing software project lifecycle and understand the role of DevOps in addressing these areas.

The advent of DevOps and its importance

In the early days of software development, the process was siloed and linear, and most companies adopted a model known as the waterfall methodology. This traditional model followed a rigid sequential order of steps: Requirement analysis, design, implementation, testing, deployment, and maintenance. Each phase was distinct, and one phase had to be completed before moving on to the next. The operations teams were largely separated from the development process, only stepping in at the deployment and maintenance stages.

These silos resulted in several issues. Developers would spend months, even years, coding without sufficient input from operations. When the code was finally passed on for deployment, operations often encountered problems that required going back to developers for fixes. This resulted in a **throw-it-over-the-wall** approach that led to conflicts, delays, and inefficiencies. Furthermore, the inflexibility of the waterfall model meant that changes in business requirements during the development process could lead to project failure.

The recognition of these problems gave birth to a new philosophy: **DevOps**. The term DevOps combines **development** and **operations**, symbolizing the breaking down of barriers between these two traditionally separate teams. DevOps encourages collaboration and communication between developers and IT professionals while automating the process of software delivery and infrastructure changes. This fosters a culture of shared responsibilities, where both teams work together throughout the software lifecycle, from design through production support.

A central principle in DevOps is the idea of **continuous integration and continuous deployment** (**CI/CD**). This approach promotes frequent code versions, which are automatically tested and prepared for release, enabling rapid iterations. By integrating regularly, developers can detect errors quickly and locate them more easily because each integration is relatively small. The essence of DevOps, therefore, is to deliver software faster, more reliably, and with fewer bugs.

The emergence of DevOps marks a significant shift in IT culture, where speed and ongoing enhancements are at the heart of its value proposition. In the modern digital era, businesses are under pressure to deliver high-quality software at a rapidly increasing pace. The speed at which technological innovations are occurring, coupled with the demand for new and enhanced digital experiences, necessitates frequent releases and rapid changes.

In this high-speed environment, DevOps provides the necessary agility and speed. Through automation, repetitive tasks are streamlined, reducing the possibility of human error and freeing up time for teams to focus on more critical problem-solving tasks. This makes the development and operations process more efficient and effective, with faster deployment times and fewer defects.

However, speed is not everything. DevOps also brings about increased resilience. In an always-on, connected world, downtime can be catastrophic. As DevOps encourages automating deployments and running tests in production-like environments, the chances of unexpected failures decrease. When failures do occur, DevOps' practices of monitoring and logging enable quicker detection and recovery, reducing downtime.

For example, take *Netflix*. They use DevOps to make sure their streaming service runs smoothly all the time. By constantly testing and updating their software, they can fix any problems before customers even notice them. This keeps people happy and coming back for more.

Another example is *Amazon*. They use DevOps to handle the massive number of orders they get every day. They can keep up with the demand without any hiccups by automating tasks like deploying new features and scaling up their servers when needed.

So, as we dig deeper into this book, we will learn more about how DevOps works and how it allows organizations to better adapt to changing business needs. By the end, you will see how DevOps is not just a choice anymore – it is a must-have for staying competitive in today's fast-moving digital world.

Fundamentals and principles of DevOps

The DevOps life cycle is a series of interconnected stages, each integral to successfully executing the DevOps methodology. This life cycle embodies a continuous loop of seven stages: Code, build, test, package, release, configure, and monitor.

The **code** stage involves the development team writing and reviewing the source code. It is where the process begins, setting the foundation for the rest of the life cycle. **Build** is the stage where the code is compiled into an executable file or a package. It is a crucial transition point from raw code to a product that can actually be deployed.

Next, in the **test** stage, the built packages undergo various tests to identify any bugs or errors. Once the software passes all tests, it moves to the **package** stage, where it is bundled with other components needed for deployment. The **release** stage is where the packaged and ready software is placed into the production environment. This can happen manually or automatically through continuous deployment practices.

After release, the **configure** stage handles any necessary changes or adjustments to the software to ensure it integrates smoothly within the production environment. Lastly, the **monitor** stage involves the continuous observation of the software to catch and address any potential issues swiftly. Each of these stages is vital and continually loops back to the start, reflecting the ongoing nature of DevOps. DevOps emphasizes:

- **Continuous integration and continuous deployment (CI/CD)**: The practice of frequently integrating code changes into a shared repository and automating the build, test, and deployment processes to deliver software rapidly and reliably.

- **Infrastructure as code (IaC)**: The process of managing and provisioning infrastructure through machine-readable definition files, promoting consistency, repeatability, and version control.

- **Automated testing**: Leveraging automation to conduct various tests, including unit tests, integration tests, and security tests, to detect bugs early and ensure software quality.

- **Cultural shift**: Encouraging communication, collaboration, and integration between development and operations teams to foster a culture of shared responsibility, learning, and continuous improvement.

Finally, integration is the linchpin that holds communication and collaboration together. By integrating their workflows, teams can align their goals, synchronize their efforts, and ensure that every stage of the life cycle is tuned to deliver value to the end-users.

In conclusion, adopting DevOps is a journey that involves understanding its life cycle, embracing its fundamental principles, and making the necessary cultural shifts. It is about striving for continuous improvement, breaking down barriers, and working together towards a common goal. By mastering these concepts, organizations can make the most out of the opportunities that DevOps presents, boosting productivity, efficiency, and product quality. Refer to the following figure illustrating the DevOps lifecycle:

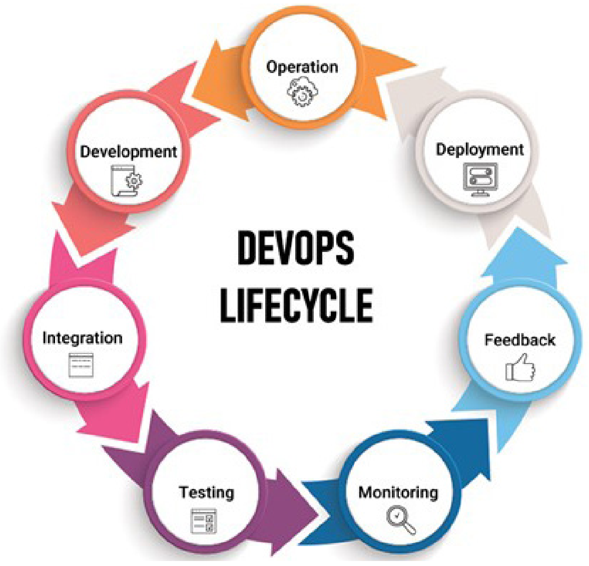

Figure 1.1[1]*: DevOps lifecycle: Key components*

Role of automation in DevOps

Automation lies at the heart of DevOps, enabling organizations to streamline processes, minimize errors, and accelerate software delivery. By automating repetitive tasks such as *code compilation*, *testing*, *deployment*, and *infrastructure provisioning*, *DevOps teams* can:

- **Increase speed and efficiency**: Automation reduces manual effort and enables faster, more frequent software releases, allowing organizations to respond quickly to market demands and customer feedback.
- **Enhance accuracy and consistency**: Automated processes ensure uniformity and eliminate human error, leading to more reliable and predictable outcomes across the software development life cycle.

1. Source: Spiceworks

- **Improve scalability and resilience**: Automation enables organizations to scale their infrastructure and applications seamlessly, adapt to fluctuating demand, and recover quickly from failures, thereby enhancing resilience and availability.

Tools and technologies for DevOps automation

There are a myriad of tools and technologies available to facilitate DevOps automation, each addressing different aspects of the life cycle. Some of them are discussed below:

- Git is a distributed version control system that allows developers to track and manage changes to their code. It enables multiple developers to work on the same codebase simultaneously without overwriting each other's changes, thereby supporting collaboration.
- Jenkins, a CI/CD tool, automates various stages of software delivery, like *building*, *testing*, and *deploying code*. By using Jenkins, teams can speed up the development process and ensure that the code is ready for deployment at any given moment.
- Docker, a containerization tool, packages an application along with its environment, dependencies, and libraries into a standalone unit called a container. This ensures that the application runs consistently across different computing environments.
- Kubernetes, another essential tool, orchestrates and manages containers at scale. It handles tasks like *deployment*, *scaling*, and *management of containerized applications*, *simplifying complex operations*.
- Ansible is an IaC tool that automates software provisioning, configuration management, and application deployment. With Ansible, the infrastructure's state is defined in a declarative language, which simplifies management and enhances consistency. Lastly, cloud platforms like *AWS*, *Azure*, and *Google Cloud* provide services that support DevOps practices. They offer a vast array of resources and services that help in automating tasks, scaling infrastructure, and improving availability and performance.

Choosing the right tool for your specific tasks and workflows is essential. While a vast array of tools exists, the choice depends on your organization's needs, existing workflows, skill sets, and budget. For instance, if your team is already using *Python*, *Ansible*, a *Python-based tool*, might be a logical choice for IaC. If your applications are containerized, Kubernetes might be indispensable. The key is to understand the strengths and limitations of each tool and how they align with your needs.

As we conclude this introduction chapter, it is important to set some expectations and prerequisites for this book. This text is intended for professionals working in software development, IT operations, or anyone interested in understanding and implementing

DevOps practices. We assume that readers have a basic understanding of software development and operations concepts. Familiarity with Linux and scripting languages would be beneficial but not mandatory.

Readers can expect to gain a comprehensive understanding of DevOps, its principles, practices, and the role of automation. You will learn about the tools and technologies that facilitate DevOps automation and how to choose the right ones for your needs.

Emerging trends in DevOps tools and technologies

DevOps is a constantly evolving field, with new tools and technologies emerging all the time to help teams work more efficiently and effectively. Let us take a look at some of the emerging trends in DevOps:

- **GitOps**: GitOps is a methodology that brings together the principles of DevOps and Git version control. It involves managing infrastructure and application deployments using Git repositories as the single source of truth. With GitOps, teams can automate deployment workflows, improve collaboration, and ensure consistency across environments.

- **Serverless computing**: Serverless computing, also known as **Function as a Service** (**FaaS**), is gaining popularity in DevOps circles. It allows developers to focus on writing code without worrying about managing servers. With serverless computing, applications are broken down into small, event-driven functions that are executed in response to specific triggers. This approach enables rapid scaling, cost optimization, and reduced operational *overhead*.

- **Observability**: Observability is becoming increasingly important in DevOps as organizations seek greater insights into their systems and applications. Traditional monitoring tools focus on metrics, logs, and traces, but observability goes beyond these to provide a holistic view of system behavior and performance. Emerging observability platforms leverage ML and AI to analyze complex data and identify patterns, anomalies, and potential issues proactively.

- **ChatOps**: ChatOps is a collaboration model that integrates chat tools like Slack or *Microsoft Teams* with DevOps workflows. It allows teams to execute commands, trigger automated processes, and access information directly from within their chat environment. ChatOps fosters real-time communication, enhances transparency, and streamlines decision-making, leading to faster problem resolution and improved productivity.

- **Infrastructure as code (IaC) enhancements**: IaC is a fundamental practice in DevOps, enabling teams to automate the provisioning and management of infrastructure using code. Emerging trends in IaC focus on improving tooling, enhancing scalability, and enabling greater abstraction and reuse of infrastructure