Cloud-Native DevOps

Building Scalable and Reliable Applications

Mohammed Ilyas Ahmed

Apress®

Cloud-Native DevOps

Building Scalable and Reliable Applications

Mohammed Ilyas Ahmed

Apress®

Cloud-Native DevOps: Building Scalable and Reliable Applications

Mohammed Ilyas Ahmed
Boston, MA, USA

ISBN-13 (pbk): 979-8-8688-0406-9 ISBN-13 (electronic): 979-8-8688-0407-6
https://doi.org/10.1007/979-8-8688-0407-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Coordinating Editor: Gryffin Winkler

Cover designed by eStudioCalamar

Cover image by Enrique from Pixabay (www.pixabay.com)

To my beloved parents, Fareeda Tabassum and Mohammed Altaf Ahmed,

Your unwavering love, guidance, and sacrifices have shaped every step of my journey. Your endless support and belief in my abilities have been a constant source of strength. I am forever grateful for the values you have instilled in me and the lessons you have taught me.

Thank you for being my steadfast pillars of strength, for teaching me resilience, and for shaping the person I am today. This achievement is as much yours as it is mine.

With heartfelt gratitude and endless love,

—Mohammed Ilyas Ahmed

Table of Contents

About the Author

Mohammed Ilyas Ahmed is an industry professional with extensive expertise in security within the DevSecOps domain, where he diligently works to help organizations bolster their security practices. With a fervent dedication to enhancing security posture, Mohammed's insights and guidance are invaluable to those navigating the complex landscape of DevSecOps. In addition to his involvement in industry events, Mohammed is an active speaker and judge, lending his expertise to technical sessions at prestigious conferences. His commitment to advancing knowledge is evident through his research contributions at Harvard University, where he contributes to journal publications, enriching the academic discourse surrounding security practices, and, as a distinguished member of the Harvard Business Review Advisory Council, underscores his commitment to advancing knowledge and fostering collaboration between academia and industry.

Mohammed Ilyas Ahmed's influence extends even further as a Member of the Global Advisory Board at VigiTrust Limited, based in Dublin, Ireland. This additional role highlights his international reach and his involvement in shaping global strategies for cybersecurity and data protection.

Mohammed's dedication to excellence is further highlighted by his numerous certifications, which serve as a testament to his proficiency and depth of knowledge in the security domain. However, beyond his professional pursuits, Mohammed is a multi-faceted individual with a diverse range of interests, adding richness to his character and perspective.

From thought to action: Grow through what you go through. Also add some spacing between my bio and this quote and justify as center

About the Technical Reviewer

Shivakumar R. Goniwada is a renowned author, an inventor, and a technology leader with more than 25 years of experience in architecting cutting-edge cloud-native, data analytics, and event-driven systems. He currently holds a position as Chief Enterprise Architect at Accenture, where he leads a team of highly skilled technology enterprise and cloud architects. Throughout his career, Shivakumar has successfully led numerous complex projects across various industries and geographical locations. His expertise has earned him ten software patents in areas such as cloud computing, polyglot architecture, software engineering, and IoT. He is a sought-after speaker at global conferences and has made significant contributions to the field through his publications. Shivakumar holds a degree in technology architecture and certifications in Google Professional, AWS, and data science. He also completed an Executive MBA at the prestigious MIT Sloan School of Management. His notable books include *Cloud-Native Architecture and Design*, *Introduction to Datafication*, and *Introduction to One Digital Identity*, all published by Apress.

CHAPTER 1

Unveiling the Cloud-Native Paradigm

> ***"Unveiling Cloud-Native: Embracing the future with digital brilliance!"***

Welcome, fearless explorer. As we embark on our journey into the Cloud-Native DevOps realm, remember our motto: "Go Native, Go Cloud!" It's not just a saying; it's a guiding principle for those who dream of taking applications to new heights, much like your caffeine levels during those late-night coding sessions.

Cloud-Native DevOps isn't just the support crew; it's the star of the tech stage, delivering a performance with the precision of a finely tuned machine. Get ready for an adventure that's not just about adaptation but a fast-paced journey into digital excellence, seamlessly creating, deploying, and scaling applications like a top-tier show on opening night.

In the ever-evolving digital landscape, businesses are leveraging cloud-native technologies to develop and deploy applications at unprecedented speeds. It's like having a powerful toolkit for creating, deploying, and scaling applications efficiently.

Containers are our versatile tools, designed to handle every shift and change in the cloud environment. Microservices are the efficient building blocks, fine-tuning our applications with perfect precision. And automation? It's the backbone of our operations, ensuring our code

M. I. Ahmed, *Cloud-Native DevOps*, https://doi.org/10.1007/979-8-8688-0407-6_1

performs brilliantly from the start. Whether you're an experienced tech professional or new to the field, this chapter promises an insightful and engaging journey through orchestration, microservices, and the fast-paced process of continuous integration and continuous delivery (CI/CD). So, prepare your code, because in the Cloud-Native DevOps arena, the only thing more impressive than our applications is the pace of innovation. Get ready to code your way to success in the dynamic world of digital innovation!

In this chapter, we will be encompassing the following foundational topics:

- Pre-cloud Era
- Evolution of Cloud Native
- Introduction and Understanding of Cloud Native

Pre-cloud Era

Before we dive into the cloud native, ever thought about how computing functioned before the cloud? Let's turn the clock back. Organizations relied on traditional infrastructure, in other words, on-prem IT infrastructure, where they used to have physical servers installed on their own building called data centers.

A number of intricate factors must be taken into account when establishing and managing a conventional data center. It is necessary to safeguard the physical space, either by acquiring new locations or by securing existing data centers. The often-overlooked electric power requirements force large-scale server array planning, make sure power conduits are adequate, and include backup generators for operational resilience. Physical security, which includes key/badged access points, surveillance tools, and security personnel, is crucial in enterprise deployments. There are obstacles associated with network connectivity, such as the need for redundant connections and possible infrastructure expansion by Internet service providers, contingent upon governmental approvals.

Because of the heat produced by equipment, cooling solutions are essential; some data centers have passive cooling systems. Last but not least, from ordering to testing, the procurement, setup, and utilization of physical hardware including network, computer, and storage components demand a substantial number of resources. The complete data center infrastructure must be designed, ordered, installed, and run simultaneously, requiring a sizable workforce.

Businesses now enjoy greater convenience because they don't have to worry about buying and maintaining servers. They can choose to save money by renting resources from cloud providers instead. With this approach, they can readily adapt their resources to meet their demands at any given time, and geographical barriers no longer limit access to data and applications.

Evolution of Cloud Native

In the swiftly changing realm of technology, the rise of cloud-native architecture has transformed the approach to developing, deploying, and managing applications. Leveraging the capabilities of cloud computing services and principles, cloud-native architecture stands as a pivotal driver for augmenting the scalability, reliability, and agility of contemporary applications. Embedded in the fundamental tenets of resilience and adaptability, this architectural paradigm serves as the foundation for pioneering technological advancements in the digital domain. Orchestration tools like Kubernetes further streamlined the deployment and management of containerized applications, marking a pivotal moment in the cloud-native landscape. As the landscape continues to evolve, organizations are poised to harness the full potential of cloud-native architectures to meet the ever-changing demands of the digital era.

Shift from Mainframe Computing to a Cloud-Native Approach

The transition from mainframe computing to a cloud-native approach is a significant trend in the IT industry, driven by the need for greater agility, scalability, and cost-efficiency. Mainframes have long been the backbone of enterprise computing, providing reliable and secure processing for mission-critical applications. However, their rigid architecture and high cost of ownership have made them less appealing in an increasingly dynamic and cost-conscious IT landscape (Figure 1-1).

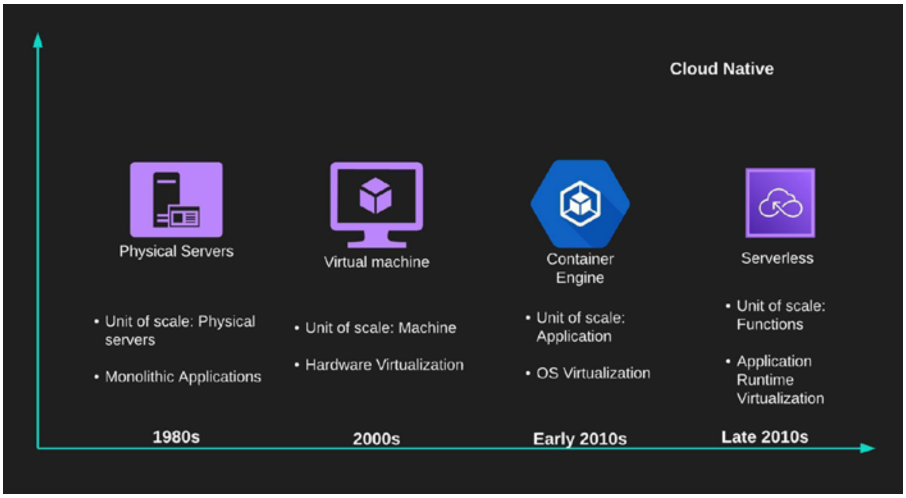
Cloud Native
Physical Servers
Virtual machine
Container Engine
Serverless
• Unit of scale: Physical servers
• Monolithic Applications
• Unit of scale: Machine
• Hardware Virtualization
• Unit of scale: Application
• OS Virtualization
• Unit of scale: Functions
• Application Runtime Virtualization
1980s
2000s
Early 2010s
Late 2010s

Figure 1-1. *From mainframe computing to a cloud-native approach*

In contrast, cloud-native computing offers a more flexible and scalable approach to software development and deployment. Cloud-native applications are designed to be lightweight, resilient, and easily deployed to cloud platforms. This makes them well-suited for the modern business environment, where organizations need to quickly adapt to changing market conditions and customer demands.

As monolithic applications became increasingly challenging to manage due to their complexity, the need for greater scalability, flexibility, and rapid deployment became apparent. In response, cloud computing emerged as a transformative force. The transition to cloud-based infrastructure opened up a plethora of opportunities for businesses to leverage scalable and cost-effective computing resources, thereby fostering innovation and facilitating the creation of responsive, on-demand applications. Additionally, in this evolving landscape, even monolithic applications are now being containerized, further enhancing their manageability and adaptability within cloud-native environments.

Advantages of Cloud-Native Computing over Mainframe

Scalability: Cloud-native applications can automatically scale up or down in response to workload changes, eliminating the need for organizations to over-provision or under-provision infrastructure resources.

Cost-Efficiency: Cloud-native applications often boast a lower total cost of ownership compared to mainframe applications. The pay-as-you-go model, where organizations pay for the resources they use, contributes to this cost-efficiency.

Agility: Cloud-native applications can be developed and deployed much faster than traditional mainframe applications because of the modular nature of cloud-native applications, with smaller components that can be independently developed and tested.

Innovation: Cloud-native computing fosters a more agile and innovative environment for software development. Cloud platforms offer access to a diverse range of tools and services, enabling the quick and easy development and deployment of new applications.

Disadvantages of Cloud-Native Computing over Mainframe

On the other hand, there are some disadvantages in transitioning from mainframe to cloud native. Migration of data can be a complex and time-consuming process. Applications on the mainframe need to be modernized before the migration as this is to restructure the code and substitute outdated components. Even though there are difficulties, the advantages of cloud-native computing are attractive, leading numerous organizations to invest in moving their mainframe applications to the cloud.

Higher Costs: Although cloud-native setups offer scalability, they can also be more expensive than mainframe systems, particularly for certain tasks. Mainframes may demand a hefty initial investment but can be more economical for consistent, large-scale workloads.

Increased Complexity: Cloud-native setups involve intricate distributed systems, microservice architectures, and containerization tools like Docker and Kubernetes. Managing and coordinating these elements can be more intricate than overseeing a single mainframe system.

Security Challenges: Cloud-native computing introduces additional security risks compared to mainframe environments. With data spread across various servers and services, there are more potential security vulnerabilities to address. Ensuring data security during transmission and storage becomes more complex.

Vendor Lock-In: Embracing cloud-native technologies often means relying on specific cloud provider services and APIs. This can lead to vendor lock-in, making it challenging to switch providers or migrate applications to different platforms in the future.

Performance Concerns: Although cloud computing offers scalability, the performance of cloud-native applications may sometimes lag behind mainframe systems, especially for tasks requiring high throughput or low latency processing. While advancements in cloud technologies may reduce this performance gap, it remains a consideration.

Data Sovereignty and Compliance Issues: Storing data in the cloud may raise concerns regarding data sovereignty and compliance with regulations, particularly in industries with stringent data protection requirements. Organizations must carefully manage where their data is stored and ensure compliance with relevant regulations.

The Twelve-Factor App

In today's digital landscape, software is commonly provided as a service, encompassing web apps or software-as-a-service. The Twelve-Factor App represents a systematic approach to constructing software-as-a-service applications. By employing declarative formats for automated setup, it streamlines the onboarding process for new developers, reducing both the time and costs associated with project integration. The establishment of a clear agreement with the underlying operating system ensures optimal adaptability across diverse execution environments. Minimizing disparities between development and production enables continuous deployment, fostering heightened agility in the software development life cycle. Additionally, the methodology showcases the ability to seamlessly scale up operations without necessitating significant modifications to tools, architecture, or development practices.

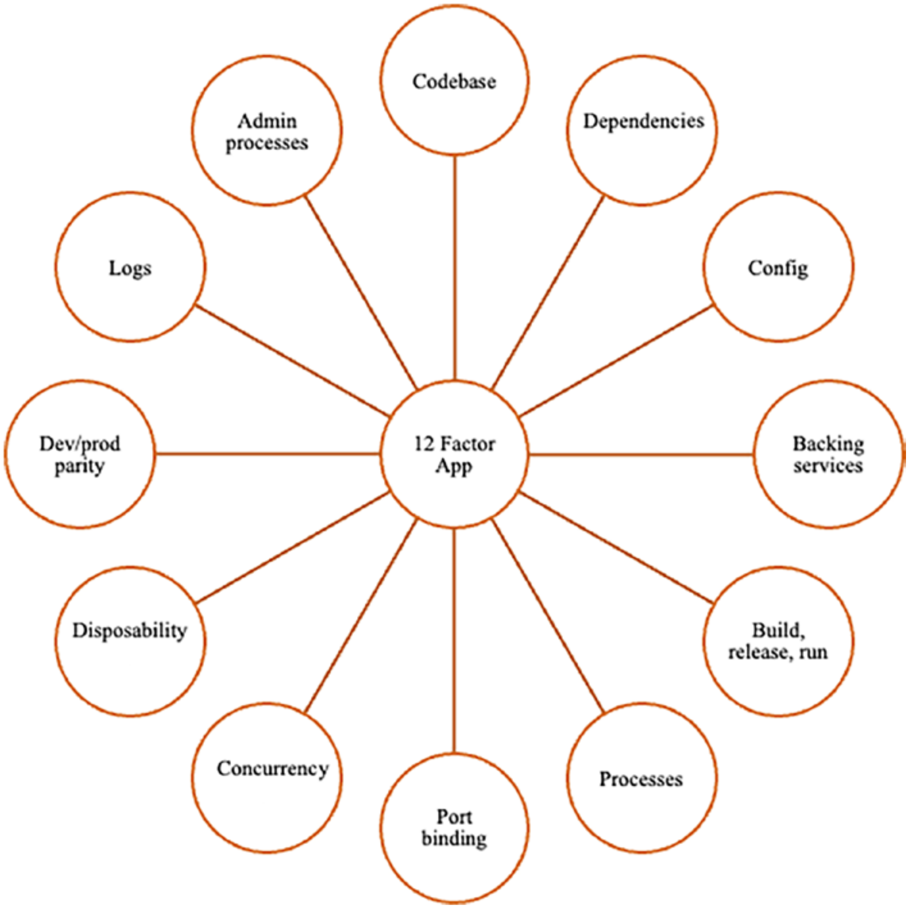

Figure 1-2. *Twelve-Factor*

Codebase: A codebase is the entirety of an application's source code, typically stored in a centralized or decentralized version control system. Each application should have a unique codebase; multiple codebases indicate a distributed system, where each component is an independent application that adheres to the Twelve-Factor App methodology. Sharing code between multiple applications violates the Twelve-Factor principle. Instead, shared code should be extracted into libraries and included through dependency management tools.

Dependencies: It comprehensively and explicitly declares all dependencies using a dependency declaration manifest. Additionally, it employs a dependency isolation tool during execution to prevent any implicit dependencies from infiltrating the surrounding system. This comprehensive dependency specification is consistently applied to both production and development environments.

Configuration: The Config principle emphasizes that configuration details should be introduced into the runtime environment through either environment variables or settings specified in a standalone configuration file. Although, in specific instances, retaining default settings directly within the code is acceptable for potential overrides, it is recommended to separate settings like port numbers, dependency URLs, and state configurations such as DEBUG. These should be maintained independently and applied during deployment. Instances of external configuration files include a Java properties file or config/database. yml file.

Backing Services: This principal advocates for architects to consider external elements like databases, email servers, message brokers, and standalone services that can be provisioned and managed by system personnel as connected resources. Examples include messaging systems – RabbitMQ – and database – MySQL.

Build, Release, Run: The Build, Release, and Run principle divides the deployment process into three distinct and repeatable phases that can be executed independently at any point in time. The Build phase involves retrieving code from the source code management system, compiling it into artifacts, and storing the artifacts in an artifact repository like Docker Hub or a Maven repository. Following the Build phase, configuration settings are applied during the Release phase. Finally, in the Run phase, a runtime environment is provisioned using scripts and a tool like Ansible, and the application along with its dependencies is deployed into the newly provisioned environment.

Processes: The Twelve-Factor App methodology advocates for stateless processes, meaning each process operates independently without maintaining state or session information. This facilitates easier scaling and prevents unintended side effects, enabling seamless addition or removal of processes to adapt to changing workloads.

Port Binding: The Twelve-Factor App methodology emphasizes using port numbers, not domain names, for service identification. Domain names and IPs can be dynamically assigned, making them unreliable references. In contrast, port numbers provide a more consistent and manageable approach for network exposure. To avoid potential port collisions, port forwarding can be employed. The port number standardization, with established conventions like port 80 for HTTP, port 443 for HTTPS, port 22 for SSH, port 3306 for MySQL, and port 27017 for MongoDB.

Concurrency: This methodology recommends organizing processes by purpose and scaling them independently to handle varying demands. As depicted earlier, an application is exposed via web servers behind a load balancer, which in turn utilizes business logic from Business Service processes behind their own load balancer. When web server load increases, that group can be scaled up separately. Similarly, if the Business Service becomes a bottleneck, it can be scaled independently. Supporting concurrency enables scaling individual application components to meet specific demands, avoiding the need to scale the entire application at once.

Disposability: This methodology emphasizes graceful application startup and shutdown. Graceful startup ensures that all necessary preparations, such as database connections and network resource access, are complete before making the application available to users. Graceful shutdown involves properly closing database connections, terminating other network resources, and logging all shutdown activities.

Dev/Prod Parity: Containers effectively package all service dependencies, reducing environment-related issues. Nevertheless, certain scenarios, particularly those involving managed services unavailable on-premises in the development environment, can be more challenging.

Logs: Streaming log data to enable access by various interested consumers. The process of routing log data should be independent of log data processing. For instance, one consumer might focus exclusively on error data, while another might prioritize request/response data. Additionally, another consumer might archive all log data for event tracking. A notable advantage of this approach is that log data persists even if the application terminates.

Admin Processes: The Admin Processes principle emphasizes that administrative tasks are integral to the software development life cycle and should be treated accordingly. As illustrated earlier, an Orders service is deployed as a Docker container alongside an admin service named Data Seeder. The data sender service is responsible for populating the Orders service with initial data.

Introduction and Understanding of Cloud Native

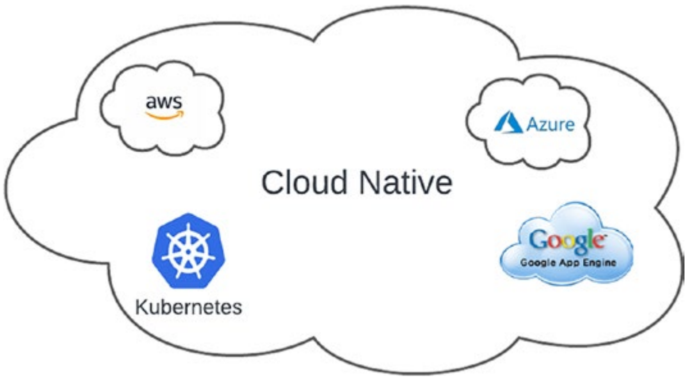

Figure 1-3. *Cloud-native fundamentals*

Cloud native has seized the spotlight in the industry and is presently creating ripples in the market. When you search for the term "What is Cloud Native?" on Google, you'll encounter numerous definitions that differ across articles. I prefer to define it as follows:

> Cloud-native is a methodology for developing and overseeing contemporary applications in the cloud. Modern enterprises aspire to construct applications capable of scaling, adapting, and swiftly updating to address dynamic customer demands. Achieving this involves employing tools seamlessly compatible with cloud systems. These technologies facilitate rapid adjustments to applications without causing service disruptions, providing businesses with a distinct competitive advantage.