International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 11 | Nov - 2022 SJIF Rating: 7.185 ISSN: 2582-3930

Advancements in Blue-Green Deployment Techniques within Cloud Native Environments

Ramasankar Molleti,
Independent Researcher, Email: sankar276@gmail.com , USA

Abstract
Blue-green deployment has become very important in ensuring that there is minimal time when the system is offline and reducing other risks associated with updating the software in cloud-native architecture. This paper aims to discuss the modern trends in blue-green deployment, focusing on the change from simple to complex structures. In this paper, the discussion on Containerization, orchestration platforms, and automated testing pipelines has greatly improved the capability of blue-green deployment. Some of the issues that are covered include the issue of managing database schema, managing the resources that are used, and the real-life scenarios that are involved. Thus, the effectiveness of the metrics for measuring different aspects of the blue-green deployment is considered, and the influence of the approach on the frequency of deployment and system stability is discussed. This paper also discusses the future of the deployment research area and the possibility of incorporating artificial intelligence into the deployment processes suggesting that further development of intelligent and strong deployment in cloud-native environments is possible.

Keywords: Blue-Green Deployment, Cloud-Native, Containerization, Orchestration, Continuous Deployment, Traffic Routing

I. Introduction

Cloud computing has been a rapidly growing concept and has influenced how software is developed and upgraded. As the usage of Cloud-Native Architectures is shifting there must be a means of deploying such architectures that is efficient and dependable. The blue-green deployment has been established to be a strong practice on how to prevent downtime, and consequently, risks that are attributable to software updates. The blue-green deployment is a technique that means the usage of two identical production environments named “blue” and “green”. This strategy assists organizations in operating effectively with versions and does not impact the audience when changes are made with new releases. Later on, cloud-native technologies were invented and accordingly, the blue-green deployment techniques were also progressed to a new level to meet the new challenges and opportunities.

This paper is used to highlight the changes that have occurred in the blue-green deployment strategies, particularly in cloud-native environments. Therefore, in the given article, the author proceeds with the description of the evolution of deployment strategies beginning from the classical ones up to the blue-green ones. Also, the paper examines major components that are the core of blue-green deployment, including the creation of the two environments, traffic redirection, and the rollback process. Therefore, it will be possible to describe various aspects of how blue-green has been optimized and tuned for the cloud-native systems and reveal what benefits and shortcomings it may involve and what further perspectives for its evolution may be expected in the software delivery.

II. Evolution of Deployment Strategies in Cloud Native Environments

Traditional Deployment Approaches

Deployment strategies’ evolution started with the ‘big bang’ method, in which whole applications were deployed at once. This often led to large periods of system unavailability and high-risk conditions especially when dealing with large systems. When software became complex, new and better ways of

implementation were looked at, thus phasing [4]. This approach involved deploying the updates in stages to the segments of the infrastructure or the users, minimizing some risks while adding the problem of having to manage several versions at the same time.

Emergence of Blue-Green Deployment

Blue-green deployment originated in the early 2010s as a new idea to be applied to cloud-native systems, which could resolve the problems that arose from the generally used models of deployment. This approach introduced a simple yet powerful concept: It is advisable to have two production lines where one is in operation and the other is idle so that the operational one can be cleaned and set for the next round. The process involves introducing a new version of the application and making sure that it runs efficiently in the non-operational mode [7]. After the rightfulness of the green web traffic is determined, the web traffic can transition smoothly from blue to green, which changes their positions.

This strategy brought immediate benefits:

1. Minimizing the downtime to nearly zero and the environment swaps can be done in a matter of seconds.
2. Reduction of the likelihood of deployment failure as fresh versions are deployed in a production environment before other versions.
3. Possibility to have an almost real-time rollback that can be utilized in case of the appearance of issues after the switch.

Blue-Green Deployment strategy continued to be implemented with the advancement of cloud-native solutions. The latest modern general-purpose container orchestration systems or platforms like Kubernetes provided the right space for these deploys as they offered very good networking and traffic management [5]. It can also be considered one of the stages that contributed to the improvement of the deployment strategy, as it excluded numerous issues associated with the use of the conventional methods. The paper shows the way for such developments since they had to be adapted to the modern cloud-native design principles and their implications.

III. Core Principles of Blue-Green Deployment

Dual Environment Setup

Blue-green deployment is one way of applying the idea of having two production environments, and its beginning involves building and maintaining two of them. These environments are traditionally called “blue” and “green” and the structure, setup, and capability of both are supposed to be identical. This duplication is necessary when it comes to the transition from one version to another and when it comes to speed. In cloud-native architecture, the dual environment setup leverages the cloud resources in as much as the elasticity and scalability of resources are in consideration [8]. When it comes to the creation of such environments, Infrastructure as Code tools are very handy because they facilitate automation and templating. Since the blue environment is held in the active state ready to fix troubles as soon as possible this automation ensures that it has the same configuration as the green environment reducing those configurations that would lead to the undesired behavior in the actual or testing environment. It also assists in performing rigorous tests on recent releases as well in the dual structure that is established. This downtime environment is also useful because it can be used to release the new version because just like a live environment it will be tested in the same way. This testing phase is crucial in identifying some issues that may not be easily identified in lower environments thus minimizing the number of issues that may be realized when the system is migrated to the new version.

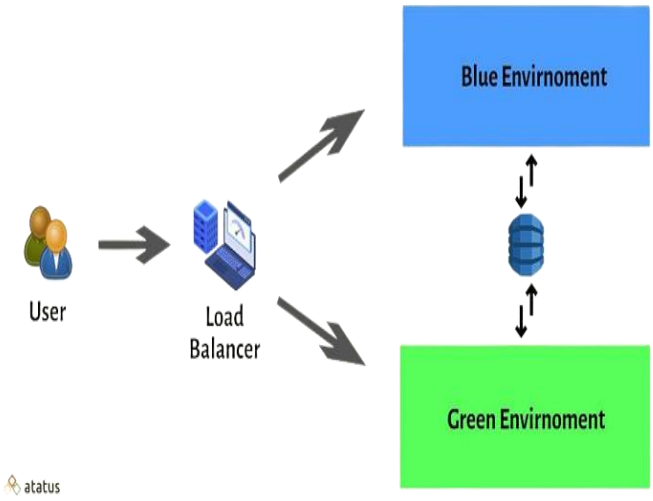

Figure 1: Blue-Green Deployment using Kubernetes

Traffic Routing and Load Balancing

Traffic routing is another fundamental aspect of the blue-green deployment model as it ensures the correct flow between the environments. While in cloud native architectures there are usually much more complex load balancing to accomplish this task. Cloud platforms of today, as well as the systems for container orchestration, allow for efficient traffic management which can be used in blue-green deployments [9]. It starts with the routing of all the production traffic to the blue environment. When the new version is deployed and tested in the green environment, the load balancer is then adjusted to route traffic progressively from the blue environment to the green. This transition can be done in several ways given the kind of application and the available infrastructure.

One such technique is the canary releases where a tiny portion of the traffic is directed to the new version. This is due to the reason that real-world testing is carried out within a limited population of users, yet useful feedback and performance data are obtained [10]. If there is no problem reported, the amount of traffic that is directed to the new version is gradually increased until total switchover. Other additional measures that can be used in traffic rerouting may include session affinity to guarantee that users are always directed to the same environment during the transition period. This is especially important in applications where the state needs to be maintained about the user.

Rollback Mechanisms

When it comes to the blue-green deployment one of the main benefits is the ability to revert to the previous version if there are issues after the switch. This capability significantly reduces deployment risks as illustrated below. The rollback in the blue-green setup takes place through traffic routing to the original environment. Since the prior version remains active and fully operational, this can be achieved within a matter of seconds minimizing the impact on the end-user's experience while at the same time making it possible to have a stable system. Traffic switching is an easier process than rollback mechanisms; the matter also includes data consistency and state [11]. Where the new version alters what that structure or content is, rollback procedures need to be able to sustain integrity. The monitoring and alarm systems should be completely operational and continually comparing the KPIs as well as the error rates with the staff in the event of any variation. If some pre-defined values are ever crossed, then there are automated rollback procedures that can be initiated that will slow things down even further. Blue-green deployment is based on the dual environment of settings, traffic redirection, and rollback. Combined, they enable more safe, low-risk deployments in cloud-native architectures, which is imperative due to the specifics of todays, distributed applications.

IV. Advanced Blue-Green Techniques for Cloud Native Architectures

Containerization and Orchestration Integration

The containerization and the orchestration platforms have been enhancing the blue-green deployments in the cloud-native architecture solutions. Containers assist in making the environments uniform, and for that reason, they are suitable for implementation in blue-green strategies. However, some solutions help to manage such deployments at scale, container orchestration systems are one of them like Kubernetes. In the case of the containerized blue-green deployments, containers are contained in the environment. The fact that containers do not change makes them part of a solid guarantee of the steadiness of environment environments and subsequent minimization of the issues that are particular to environments. Orchestration tools help in as far as the creation and destruction of these environments within the shortest time possible and are hence suitable for frequent and dependable deployment [12]. Kubernetes today is an essential part of the blue-green configurations of today's applications. Therefore, it can be concluded that both the Deployments and Services concepts can be associated with the blue-green strategy. There are several possibilities, but one of the most frequently used cases is to create two Deployments named 'blue' and 'green,' and use the Service to route the traffic. Kubernetes has a rolling update strategy that can sometimes be used for gradual transitions sometimes called rainbow deployments which are more flexible and can also accommodate more complex tests.

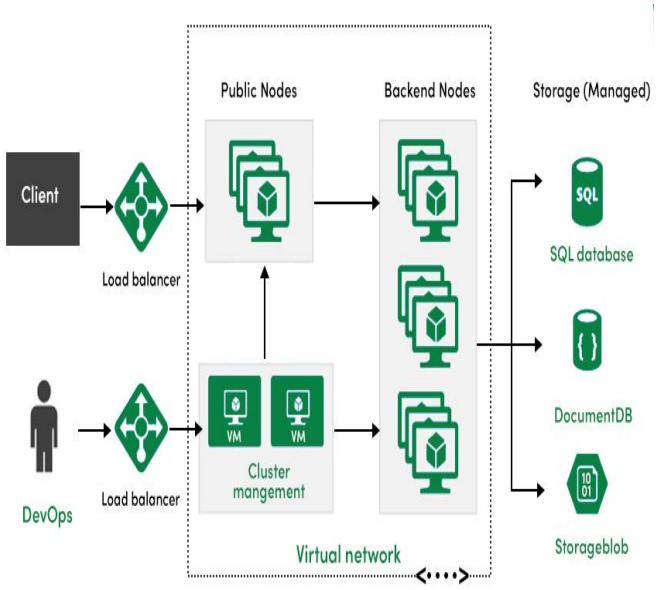

Figure 2: Containerization Integration

Automated Testing and Validation Pipelines

The new version is introduced into the blue-green deployments through automated testing or validation to check if the new version is capable of meeting the required quality and performance that is needed for the new version before it has to deal with the production traffic. The testing pipeline typically includes:

1. Unit and integration tests
2. Smoke tests for basic functionality
3. Performance tests
4. Security scans
5. Chaos engineering tests for system resilience

Higher-level methods may use artificial intelligence combined with machine learning to monitor the outcomes of tests and conclude that something is amiss that can otherwise not be detected. They are effective in identifying even such conditions as slight performance decreases or some problems with the overall GUI [13]. Another aspect of validation is synthetic user testing, which is a simulation of website utilization in the green environment before the real traffic. This supports the effort to prevent a particular issue from arising in the first place.

Database Schema Management and Data Synchronization

The main challenge of blue-green deployment in cloud-native architectures is the question of database schema management and data replication between the environments. This is accomplished by proper management of the schema and data synchronization techniques in the advanced approaches coupled with scheduling. One of them is the incorporation of the backward and forward compatibility of the database schemas for easier updates. The migration tools that enable applying partial and, at the same time, reversible schema modifications provide the opportunity to update the target schema with more granularity and accuracy and, therefore, to roll back the schema to its previous state if needed. For high-write volume applications, mechanisms that replicate data in real-time comprise change data capture (CDC) tools that help update the green environment's database [14]. A few of them use the technique of "shadow writes" where the data is written to both blue and green databases during the transition while the technique is likely to give rise to conflicts that have to be resolved. These techniques in managing data make the blue-green deployment more stable, frequent, and safe, allowing the organizations to update the software as frequently as possible without any risk of failure.

V. Overcoming Implementation Challenges

Infrastructure Scaling and Resource Optimization

The overall process of employing blue-green deployments in a cloud-native environment could be resource-consuming as it is the means of having a fully duplicated production environment during the process of deployment. This leads to an increase in costs and problems with resource management in organizations. Due to this challenge, the organizations have had to adopt what are referred to as dynamic scaling strategies. It allows one to increase the resources for a green environment by many folds during deployment and testing and then reduce them once the migration process is complete [15]. This way, there is effective management of resources and as for the costs, they are also kept under check. Other resource optimization methods also include the use of spot instances referred to as pre-emptible Virtual Machines in the green environment except for the critical ones. These lower-cost, non-critical resources are adequate for checking only and the move to more permanent resources is only made if necessary.

Metric	Before Implementation	After Implementation	Improvement (%)
Deployment Success Rate (%)	85	98	15
Mean Time to Recovery (MTTR) (minutes)	60	10	83
User Satisfaction Score	7.5/10	9.5/10	27
System Availability (%)	99.5	99.99	0.49

Table 1: Key Performance Metrics Before and After Blue-Green Deployment

Application Design Considerations

The Blue-Green architecture has to be integrated into the application that is to be deployed using the above-mentioned strategy. This may include; having loose coupling and statelessness which is very hard to implement for an application that has many combined components. To address such problems, it is becoming possible for organizations to adopt micro services architectures. There is the blue-green deployment where applications are split into minute services that can be deployed individually and thus the risk and the level of complication are minimized. One more factor that should be considered is the management of the long-term processes and the background tasks. Other solutions are even more sophisticated: These processes must not fail during environment changes and restart themselves.

DevOps Culture and Practices

The first concern arising from the advanced blue-green deployments is the cultural one. It involves a change of attitude and processes in the development, operation, and other business departments. This is best solved by having a good DevOps culture which is a synergy of development, operations, and other related functions without much repetition of tasks but rather efficient automation. This includes Development, operations, and quality assurance cross-functional teams, Automation driven from development to deployment, and the use of infrastructure as code [17]. Other such practices include the routine exercises of the deployment and rollback scenarios that need to be conducted now and then and the company culture that must embrace failures in deployments. If the above-discussed cultural and organizational issues are addressed, then firms can leverage the efficiencies of innovative blue-green deployment practices in cloud-optimized applications resulting in more consistent, more frequent, and less risky software releases.

VI. Performance Analysis and Case Studies

Metrics for Evaluating Blue-Green Deployments

Some of the performance measurements for blue-green deployments in cloud-native conditions are very important. These metrics are used with the motivation of quantifying various characteristics of the deployments in a bid to improve the performance of the deployments. Some of the measures are the time required to install the OS, the time required to provision the green environment, the time used by the tests, and traffic switching time. Deployment time is negatively related to the frequency of deployment; therefore, it shows the extent of DevOps adoption [18]. Key error rates are failed deployments, errors after the application has been deployed, and errors in the newly launched blue environment; a blue-green strategy should offer a lower error rate than the conventional approaches. Others are the response time of the application, the number of throughputs per set time, and the log of the complaints by the users which in the deployment phase should be minimal.

Metric	Traditional Deployment	Blue-Green Deployment	Improvement (%)
Deployment Time (minutes)	120	30	75
Downtime (minutes)	60	0	100

Rollback Time (minutes)	30	5	83
User Impact (%)	10	0	100

Table 2: Performance Analysis of Blue-Green Deployment

Real-world Implementation Examples

Case Study 1: E-commerce Platform

A big online store adopted the blue-green deploy approach for its cloud-based architecture to reduce the time its application was unavailable during updates. They employed Kubernetes for the container orchestration, and for the traffic routing, they developed their solution based on the Istio service mesh. Their implementation was to create two replicas of the given environment in two different Kubernetes namespaces. Database updates were controlled using the schema versioning approach and the home-grown data synchronization application. The company recorded a 99 percent compliance with the foregoing guidelines [1]. From the research, it can be concluded that the changes led to a 99% reduction in deployment-related downtime and a 200% increase in the frequency of deployments. One of the issues they had was related to services and, more specifically, the shopping cart service. To overcome this they instituted a distributed caching layer that could easily replicate between blue and green environments during the switchover.

Case Study 2: Financial Services Application

In the financial sector, there is a firm that used blue-green deployments to the trading application that is located in the cloud. For the container orchestration, the service is ECS in AWS while Route 53 is AWS's DNS-based traffic routing service. Their approach was to have two parts of ECS task definitions blue and green, and utilize Route 53 latency-based traffic routing to slowly switch the traffic. In the database activities, the 'shadow writes' technique was used to ensure that the data integrity is preserved during migration [2]. The company was also able to find out that cases of deployment risk were cut by up to 75% and, at the same time, the rates of feature deployment doubled. However, they faced a high cloud cost since they had the development environment along with the testing environment. They managed this by having a better usage of the spot instances and developing better standards to grow the company.

VII. Future Trends and Research Directions

The following trends for blue-green deployment and areas of research are likely to affect the future deployment of this concept in cloud-native environments. One of the directions is the usage of artificial intelligence and machine learning in the deployment process. These could be used to forecast possible problems in the deployment, and possible traffic flow problems during the transition, and even the rolling back process could be done automatically based on performance data. The second topic is the improvement of the strategies related to the data management for the blue-green pattern but the current approaches remain underdeveloped [3]. This means that the more complex schema evolution techniques and the real-time data propagation techniques must be able to handle the more complex and larger sets of data. The same work is also being done for blue-green deployment principles to be extended for edge computing settings as well. Due to the growing popularity of edge applications, there is a need for efficient methods on how to manage the updates in the distributed edge nodes and maintain consistency and availability. These trends show that in the future the blue-green deployments will be even more automated intelligent and to various cloud-native architectures.

VIII. Conclusion

Blue-green deployment is among the best practices used to manage update situations in cloud settings, and it has several benefits such as a shorter time when a system is not active, fewer risks that are associated with updating a program, and greater update rates. Over the years, the practices in blue-green deployments have been improved due to better practices in the deployment strategies, the use of containers and orchestration, and the improvement of testing and validation processes. Through implementing these advanced deployment strategies, and the corresponding cultural shifts, it becomes possible to enhance an organization's capability to deliver application updates rapidly, securely, and efficiently within the scope of cloud-native environments.

IX. References List

Journals

[1] Mohd Satar, N.S., Dastane, D.O. and Ma'arif, M.Y., 2019. Customer value proposition for E-Commerce: A case study approach. International Journal of Advanced Computer Science and Applications (IJACSA), 10(2), pp.454-458.

[10] Pham, P.P. and Perreau, S., 2003, March. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428) (Vol. 1, pp. 251-259). IEEE.

[11] Elnozahy, E.N., Alvisi, L., Wang, Y.M. and Johnson, D.B., 2002. A survey of rollback-recovery protocols in message-passing systems. ACM Computing Surveys (CSUR), 34(3), pp.375-408.

[12] Al Jawarneh, I.M., Bellavista, P., Bosi, F., Foschini, L., Martuscelli, G., Montanari, R. and Palopoli, A., 2019, May. Container orchestration engines: A thorough functional and performance comparison. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[13] Arachchi, S.A.I.B.S. and Perera, I., 2018, May. Continuous integration and continuous delivery pipeline automation for agile software project management. In 2018 Moratuwa Engineering Research Conference (MERCon) (pp. 156-161). IEEE.

[14] Cho, J. and Garcia-Molina, H., 2000. Synchronizing a database to improve freshness. ACM sigmod record, 29(2), pp.117-128.

[15] Dougherty, B., White, J. and Schmidt, D.C., 2012. Model-driven auto-scaling of green cloud computing infrastructure. Future Generation Computer Systems, 28(2), pp.371-378.

[16] Okoli, C. and Pawlowski, S.D., 2004. The Delphi method as a research tool: an example, design considerations and applications. Information & management, 42(1), pp.15-29.

[17] Davis, J. and Daniels, R., 2016. Effective DevOps: building a culture of collaboration, affinity, and tooling at scale. " O'Reilly Media, Inc.".

[18] Yang, B., Sailer, A., Jain, S., Tomala-Reyes, A.E., Singh, M. and Ramnath, A., 2018, July. Service discovery based blue-green deployment technique in cloud native environments. In 2018 IEEE international conference on services computing (SCC) (pp. 185-192). IEEE.

[2] Debreceny, R., Lee, S.L., Neo, W. and Toh, J.S., 2005. Employing generalized audit software in the financial services sector: Challenges and opportunities. Managerial Auditing Journal, 20(6), pp.605-618.

[3] Hagemann, G., Michaels, J., Minnice, P., Pace, D., Radin, S., Spiro, A. and West, R., 2010. ITS technology adoption and observed market trends from ITS deployment tracking (No. FHWA-JPO-10-066). United States. Joint Program Office for Intelligent Transportation Systems.

[4] Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J. and Bohnert, T.M., 2017. Self-managing cloud-native applications: Design, implementation, and experience. Future Generation Computer Systems, 72, pp.165-179.

[5] Balalaie, A., Heydarnoori, A. and Jamshidi, P., 2016. Migrating to cloud-native architectures using microservices: an experience report. In Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers 4 (pp. 201-215). Springer International Publishing.

[6] Rodriguez-Sanchez, M., 2015. Cloud native Application Development-Best Practices: Studying best practices for developing cloud native applications, including containerization, microservices, and serverless computing. Distributed Learning and Broad Applications in Scientific Research, 1, pp.18-27.

[7] Dornan, M., Morgan, W., Newton Cain, T. and Tarte, S., 2018. What's in a term?"Green growth" and the "blue-green economy" in the Pacific islands. Asia & the Pacific Policy Studies, 5(3), pp.408-425.

[8] Sisto, D., 2018. The influence of collaborative governance processes on the performance of Blue Green Infrastructure projects in the maintenance phase within Dutch cities. Interreg North Sea Region Erasmus Universiteit Rotterdam.

[9] Popa, L., Rostamizadeh, A., Karp, R., Papadimitriou, C. and Stoica, I., 2007, September. Balancing traffic load in wireless networks with curveball routing. In Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (pp. 170-179).