International Journal Of Core Engineering & Management
Volume-6, Issue-09, 2020 ISSN No: 2348-9510

INTEGRATING DEVOPS FOR EFFICIENT INFRASTRUCTURE MANAGEMENT ON AMAZON WEB SERVICES USING AWS CLOUD FORMATION AND LAMBDA

Satyadeepak Bollineni
Senior DevOps Engineer
Databricks
Texas, USA
deepu2020@gmail.com

Abstract

The study looks at how DevOps ideas can be used with AWS Lambda and CloudFormation to improve the management of cloud systems. It uses a qualitative method by reading academic papers and talking to experts in the field to look at the pros, cons, and benefits of adding these tools to a DevOps system. In AWS services, continuous development and tracking are not just important but crucial. Self-healing infrastructure and automatic infrastructure supply are also essential things to consider. The link has advantages like automation, stability, and operating efficiency. However, some things could be improved, like the need for skills and the system's complexity. Comparison to more traditional methods shows that it is more flexible, repeatable, and scalable.

Keywords— Cloud Computing, Amazon Web Services (AWS) Cloud Formation, Amazon Web Services (AWS) Lambda, Infrastructure as Code, Serverless Computing, Cloud Infrastructure Management, security, Web services, Continuous Integration, Continuous Deployment

I. INTRODUCTION

DevOps, a word formed by combining the words 'development' and 'operations,' is a flexible approach to managing software development and IT operations [1]. Its usage is targeted at providing greater efficiency of software deliveries, improved product quality, and general work organization. As computing operations have shifted to the cloud, DevOps' best practices have shown their adaptability to the challenges of the cloud environment. AWS, an industry giant in cloud solutions, offers a range of services and tools that can support most of the processes involved in the DevOps approach. This adaptability should reassure and instill confidence in IT professionals and software developers working in this field.
AWS infrastructure management should thus be highly effective for performance enhancement, cost containment, and security in an organization's cloud systems [2]. Two such services that enable this are AWS Cloud Formation and AWS Lambda. CloudFormation provides developers

International Journal Of Core Engineering & Management
Volume-6, Issue-09, 2020 ISSN No: 2348-9510

with a service similar to code-provisioning, which can be versioned and deployed repeatedly. At the same time, Lambda offers serverless computation capacity so developers can execute code without servers. This paper investigates how DevOps practices, combined with AWS CloudFormation and Lambda, can help better manage infrastructure, scale the cloud infrastructure, and obtain operational flexibility.

II. LITERATURE REVIEW

Studies on DevOps have garnered significant attention in the last decade and focused on applying its principles and practices to the lifecycle of software development and organizations. As identified in [3], there are five critical dimensions of DevOps: Partnership, involvement, quantification, reciprocal interchange, and consistency and contact. Of these principles, the most important are primarily focused on breaking barriers between development and operation teams and delivering a culture [3]. CI/CD, Infrastructure as Code (IaC), automation testing – The most basic principles of DevOps are also some of the most fundamental for delivering software in terms of volume and quality.
Cloud infrastructure management, which has emerged as a common area of interest, has stirred various concerns since the introduction of DevOps, especially concerning automation and scalability. Of interest are Security, costs, and constancy in variability between settings [4]. It has also investigated how existing DevOps practices can address these issues, such as the approaches to continuous delivery of cloud-native applications [5]. AWS CloudFormation is widely used to orchestrate IaC within AWS environments since it allows the initiative to express infrastructure variations and various other configurations in declarative versions [6]. Its applications include basic and cross-border applications. Some of the critical findings explained in the superior study include Security, cost, and consistency in cloud management.

A. AWS Lambda features and serverless computing:

AWS Lambda and associated serverless architectures have been analyzed over the last few years because they reduce infrastructural intricacy. Vishal began with the fundamentals of serverless computing and then went to the technology’s effectiveness in terms of scale and costs. Lambda is also event-based and designed to be used with other AWS services to respond to infrastructure events [7]. One of the common uses of Lambda functions is log analysis, automated backups, and security monitoring, demonstrating its applicability for ‘DevOps’ for infrastructure management [8].

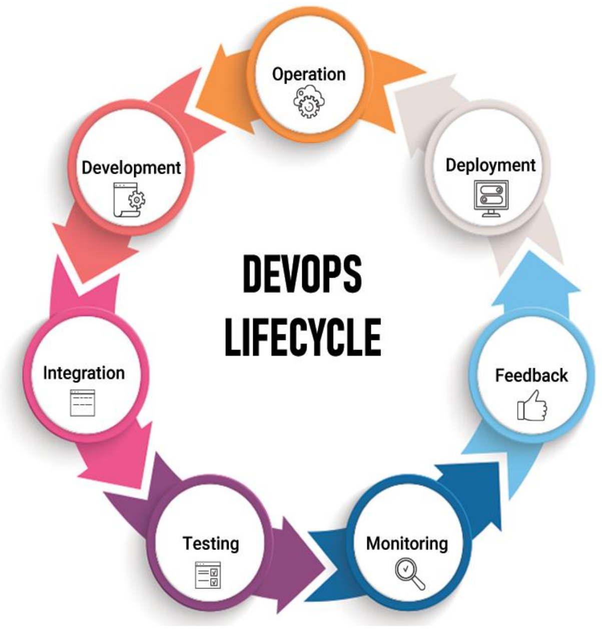

Figure 1: DevOps Lifecycle

III. METHODOLOGY

A. *Research approach*

The study will begin with a qualitative research method, focusing on professional DevOps techniques built into AWS CloudFormation and AWS Lambda. Effective infrastructure management requires these actions. The links between different DevOps parts and, more specifically, AWS features can be studied in detail with qualitative study.

B. *Data collection methods*

They involve a literature search of research articles, technical papers, case studies, and regional and international reports. Moreover, there was an interview with DevOps specialists and AWS solution architects to understand the real-world use of the described approaches and methods. These interviews were based on CloudFormation and Lambda usage in DevOps environments, problems encountered, and options for effective incorporation.

C. *Analysis techniques*

The collected data was analyzed using different thematic analysis approaches. This involved filing the interview transcripts and notes from the literature review to identify emerging patterns and themes related to DevOps practices, Cloud Formation, and AWS Lambda. The research sought to identify frequently used approaches, issues, and advantages in exploiting the technologies for cloud infrastructure management. The findings were then used to understand

International Journal Of Core Engineering & Management
Volume-6, Issue-09, 2020 ISSN No: 2348-9510

how CloudFormation and Lambda can be used elaborately in a DevOps model to improve infrastructure management practices in AWS systems.

IV. AWS CLOUDFORMATION FOR INFRASTRUCTURE AS CODE

AWS CloudFormation is a valuable service that allows developers and system administrators to define and control an infrastructure as code. It gives an imperative/declarative way of defining and creating AWS computing resources in an environment and launching them in other environments [9]. In CloudFormation, users can use templates describing system resources that need provisioning, such as EC2 instances, VPCs, S3 buckets, etc. In this approach, most steps that are usually done manually are omitted, which cuts on errors. It makes version control, which is generally referred to as infrastructure definitions, easier.

The fundamental components of Cloud Formation are a template with several features. Usually, templates are written in JSON or YAML format and consist of parameters, mappings, conditions, resources, and outputs sections [10]. The most important one is the "Resources," which contains users' definitions of the AWS resources to be created and their properties. Parameters enable the creation of the input values to be changeable, while mappings enable the creation of values depending on the region or the environment. Conditions allow leveraging to create resources by given guidelines, and outputs allow users to obtain details about the created accounts for utilization in other stacks or systems.

There are several advantages of employing CloudFormation to manage infrastructure. It helps to improve consistency when setting the development, the staging, and the production environments. Template version management allows for tracking infrastructure changes' provisions and is helpful to multiple users. The supporting features include rolling updates and rollbacks for making safer and more secure deployments and quick fixes of mistakes. Some of the best practices involve shifting to templates as code modules, using a stack within a stack in a density architecture, and using some of the intrinsic AWS functions in creating resources as variables. However, some disadvantages include learning the syntax of the CloudFormation templates and the fact that the dependencies between resources must be well planned for in a DevOps environment.

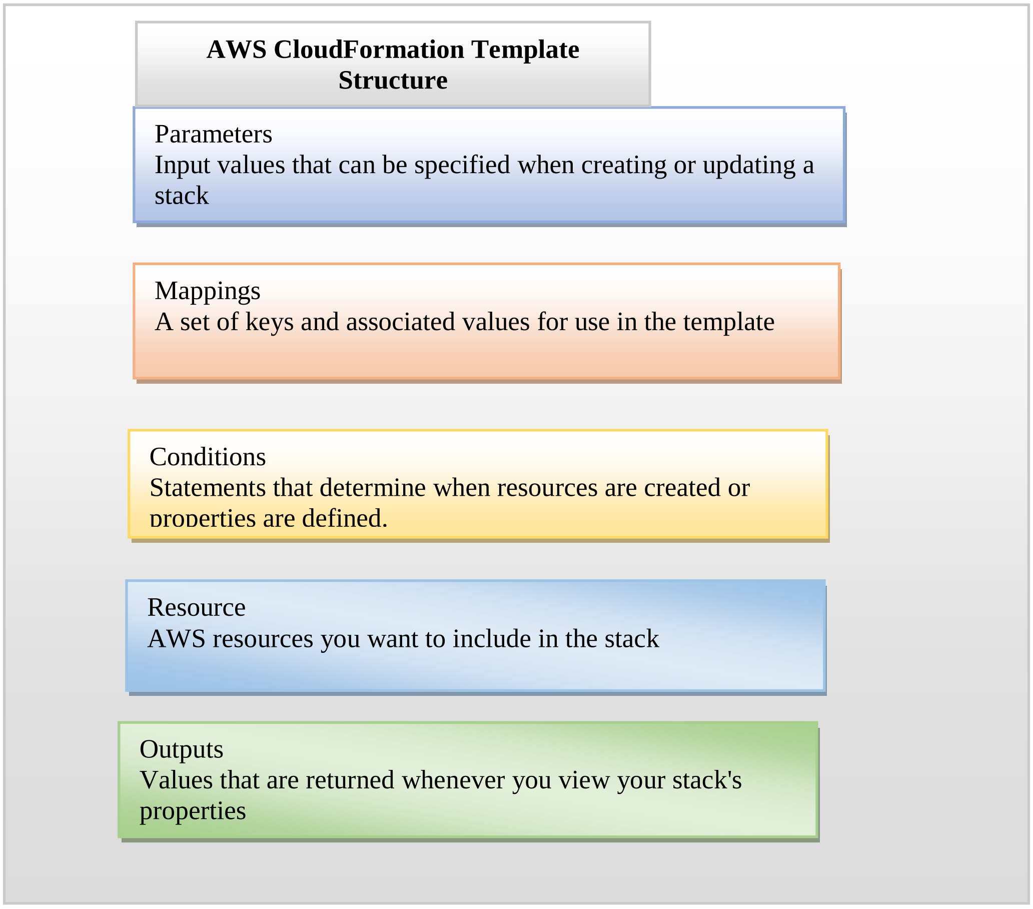
AWS CloudFormation Template Structure
Parameters
Input values that can be specified when creating or updating a stack
Mappings
A set of keys and associated values for use in the template
Conditions
Statements that determine when resources are created or properties are defined.
Resource
AWS resources you want to include in the stack
Outputs
Values that are returned whenever you view your stack's properties

Figure 2: AWS Cloud Formation Template Structure

V. AWS LAMBDA FOR SERVERLESS COMPUTING

AWS Lambda is one of the significant shifts in the model of AWS cloud computing since it provides an execution environment for code but does not require servers to be provisioned. Lambda functions are computing services triggered by events, are stateless, and can be developed in any supported programming language such as Python or Node. Js, Java, and Go, among others [11]. All these functions are intended for some particular work and can work together with the increased flow of requests or processes. Because Lambda is a serverless architecture, there is no need for activities associated with infrastructure management, which simplifies the work of the developers and allows them to devote themselves to the writing and implementation of code.

This event-driven design of Lambda has provided it with the level of flexibility and integration it needs. As described in the research, Lambda functions might be invoked by almost any event originating from other AWS services, for instance, when changes in S3 buckets, updates in the DynamoDB tables, or new messages in the SQS queues occur. Such architecture suits the development of responsive and scalable applications and readily adapts to environmental changes. For instance, a Lambda function may include some logic to process the uploaded images, send notifications, or update the records in the databases, if there are any, without requiring a permanent server.

Lambda, I have found to be more helpful in building large distributed systems because it interfaces perfectly with other AWS services. An often-overlapping service with API Gateway, it can create serverless APIs with S3 and Cloud Front for dynamic content generation or with Step Functions for advanced connectivity. Regarding infrastructure management, Lambda integrates with services such as EC2, RDS, or CloudFormation to perform tasks automatically, monitor the resources, and address operational events.

Lambda can be put to several practical uses in managing infrastructures, and the following are some of the use cases. It can be used for auto backup, log analysis, and security of resources in Amazon Web Services. Lambda functions can also include maintaining tasks, such as turning on or off the EC2 instance, considering a usage rate. Concerning DevOps, Lambda can be specifically employed to perform deploys, execute tests after a deployment, and make verifications. That's especially valuable for building self-healing infrastructure – Lambda functions can handle the events detected by the CloudWatch alarms or other monitoring systems and correct some problems themselves, increasing availability and minimal human intervention in infrastructure.

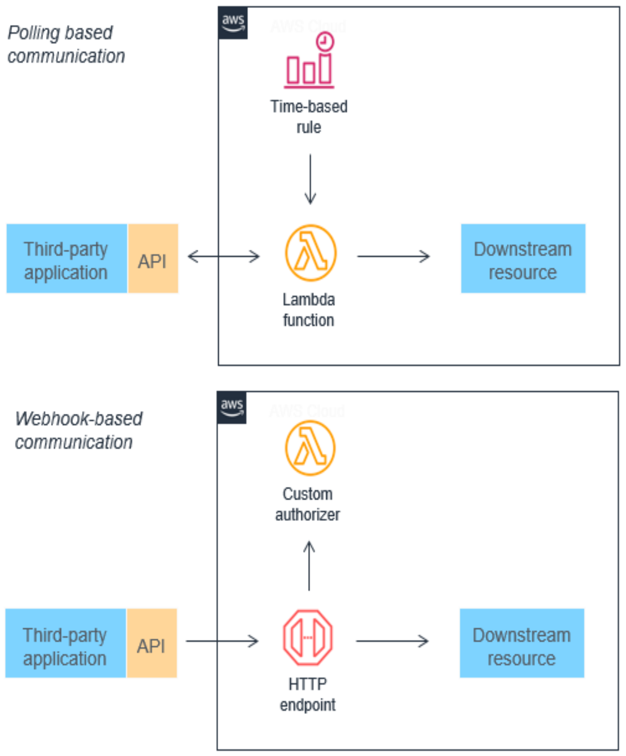

Figure 3: Lambda Architecture [11]

VI. INTEGRATION OF CLOUDFORMATION AND LAMBDA FOR DEVOPS

Here, the combination of CloudFormation and Lambda is beneficial for increasing the automation of infrastructure provisioning. The desired state can be defined by CloudFormation templates, whereas Lambda functions can be used to add definition and customization to the provisioning process [12]. For example, the Lambda function can be triggered right after the CloudFormation stack's creation to perform more configuration initiatives, set up monitoring alarms, or create databases. This integration enables the design of dynamic infrastructures that can be put in place regarding the needs of an application and the organization's needs.

What happens, however, if one delves deeper into both CloudFormation and Lambda is that the combination of the two refines the CI/CD pipelines. Lambda functions can be set at the various stages of the CI/CD pipeline to perform a test or scan, for instance, or to modify a deployment specification. Due to this, CloudFormation can be directly utilized in pipelines to ensure the versioning and deployment of infrastructure changes along with the application code, making it possible to deliver code and infrastructure continuously.

Integrated use of CloudFormation and Lambda enhances the monitoring and alerting system. Monitoring comprises a set of CloudFormation templates for CloudWatch alarms and dashboards and a set of Lambda jobs that process those alarms in real-time. For example, a

Lambda function may read through CloudWatch logs, find that there have been some irregularities, initiate alerting, or prescribe corrective actions. This integration makes the monitoring solutions much more complex and far better at addressing the needs of evolving infrastructure and applications.
It is now possible for infrastructure to self-heal when CloudFormation integrates with Lambda. CloudFormation guarantees that the correct image of infrastructures is described accurately and unambiguously, and Lambda functions allow for permanent observing of their variations and subsequent correction. An EC2 instance in a non-responsive state can be stopped by a Lambda function and replaced through a Cloud Formation update process to avoid much-unrequired intervention to maintain infrastructure stability and standards of set specifications.

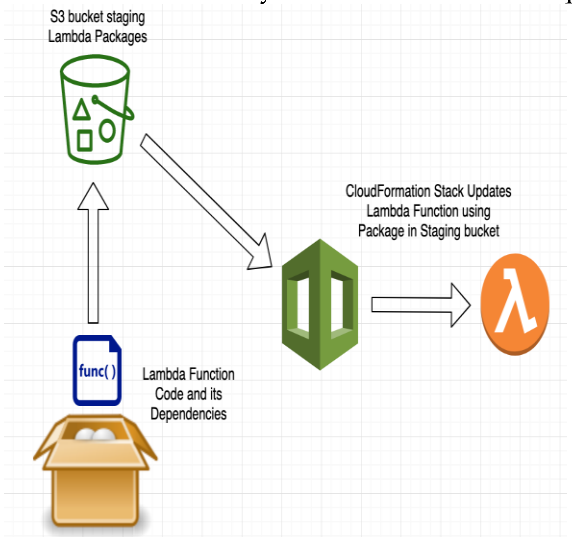

Figure 4: Integration of Cloud Formation and Lambda [12]

VII. DISCUSSION

A. Benefits and challenges of the integrated approach

CloudFormation and Lambda for DevOps practices in AWS environments have several advantages, like automation, consistency, and optimization. This approach, therefore, allows the management of large infrastructures in the organization with minimal human interference and chances of errors. However, executing this integrated approach is difficult because of the following factors; for example, the planners and developers require skills in the infrastructure-as-code model and serverless computing. Organizations must harness these technologies by training the employees and purchasing the right tools and equipment. Third, there is the matter of coping with greater levels of distributed systems and implementing the appropriate security measures in automatic processes.

B. *Comparison with Traditional Operational Methods of Infrastructure Management*

Aspect	*Traditional Approach*	*Cloud formation + Lambda approach*
Infrastructure	Manual Configuration	Infra as code (IaC)
Management	GUI or CLI	Version-controlled
Deployment Time	Hours to Days	Minutes to Hours
Scalability	Manual Scaling	Auto-Scaling
Cost-efficiency	Potentially wasteful	Pay-per-use
Consistency	Prone to human error	Reproducible environments
Automation	Limited, script-based	Event-Driven, serverless

Table 1: Comparison of traditional and CF-Lambda approach

Compared with traditional infrastructure management approaches, using CloudFormation and Lambda integration offers several benefits, such as flexibility and the possibility of reproducing the solution. In this respect, conventional approaches commonly employ configuration and scripting that involves routing a set of scripts through a console, which is error-prone and not easily scalable. The integrated approach enables infrastructure definition changes to be maintained in a version-controlled manner, enables automatic deployment, and allows changes to be made quickly [12]. It also helps improve communication between the development and operations teams, as is the general concept of DevOps. While this approach would work, it may need a change of culture and practice, which some organizations may need help implementing, which forms a barrier.

C. *Scalability and cost-efficiency considerations*

This integrated approach is also scalable; hence, the cost of managing it is also low. While CloudFormation can automate infrastructure replication in multiple regions or across numerous accounts, a simple button press can scale an application [12]. Lambda provides an optimal serverless architecture in that the resources are drawn when needed and cheaper than always-on servers. However, it would help if you designed architecture to avoid ending up with the bill shock, especially with a high frequency of invocations of Lambda or CloudFormation with the nested structure. The long-term maintainability of infrastructure-as-code and serverless functions within the organization ensures it is still comprehensible as the system becomes more extensive and complex.

VIII. CONCLUSION

DevOps of AWS CloudFormation and Lambda hold great synergy when utilized to manage the infrastructure based on cloud systems.
This research clearly shows that employing all these technologies can yield the benefits of high levels of automation, scalability, and flexibility in operations.
This knowledge is vital as the practice of DevOps transforms. Much research is required on emerging integration patterns, security, and other issues associated with becoming complex multi-server structures when delivered as AWS offerings.

REFERENCES

1. Erich F. M. A., Amrit C., and Daneva M. 2017. A qualitative study of DevOps usage in practice, Sep 2017.
2. Kaur, G. Raj, S. Yadav and T. Choudhury, "Performance Evaluation of AWS and IBM Cloud Platforms for Security Mechanism," 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Belgaum, India, 2018, pp. 516-520.
3. Welder Pinheiro Luz, Gustavo Pinto, and Rodrigo Bonifácio, “Building a collaborative culture,” Oct. 2018, pp. 14-15.
4. Ciprian Cr˘aciun “Building blocks of scalable applications,” July 2012, pp. 08-10.
5. Banijamali, P. Jamshidi, P. Kuvaja, and M. Oivo, “Kuksa: A Cloud-Native Architecture for Enabling Continuous Delivery in the Automotive Domain,” Product-Focused Software Process Improvement, Nov 2019, pp. 455–472.
6. Chen, Lianping. “Microservices: Architecting for Continuous Delivery and DevOps.” Dec 2018, pp. 1-3.
7. Erich F. M. A., Amrit C., and Daneva M. “A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process”, 2017, pp. 6 – 29.
8. Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., Tamburri, DevOps: Introducing Infrastructure-as-Code., 2017., pp. 497-498.
9. Wettinger, J., Andrikopoulos, V., Strauch, S., Leymann, F. Characterizing and Evaluating Different Deployment Approaches for Cloud Applications, 2014 pp. 5-20.
10. W. H. Siers, “Generating specifications for JSON APIs for formal and semantic use,” 2019, pp. 4-6.
11. S. Shrestha, “Comparing Programming Languages used in AWS Lambda for Serverless Architecture,” 2019, pp. 8-12.
12. M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, “Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions,” Future Generation Computer Systems, Nov. 2017, pp. 8-10.