International Journal of Computer Sciences and Engineering
Vol.**11**, Issue.**9**, pp.**11-16**, September **2023**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Paper

Efficient Resource Utilization with Auto Tagging Using Amazon's Cloud Trail Services

Sai Teja Makani[1]

[1]IT & Cyber Security Department, Spotter INC, Allentown, USA

Author's Mail Id: saitejamakani@gmail.com

Received: 18/Jul/2023; **Accepted**: 22/Aug/2023; **Published**: 30/Sept/2023. **DOI**: https://doi.org/10.26438/ijcse/v11i9.1116

Abstract: In the realm of resource management, the practice of labeling, ledgering, and tagging has a rich historical legacy that has transcended time, demonstrating its enduring importance. These processes, which have been fundamental since ancient times, continue to wield immense significance in the contemporary context, particularly when applied to intangible assets, which are instrumental in organizational success. However, in the contemporary landscape characterized by digitalization and the proliferation of non-tangible assets, the task of effectively labeling and tagging resources has grown markedly intricate. This complexity is especially conspicuous when considering intangible resources, and it is accentuated within the domain of software infrastructure and application modules. In these domains, the sheer volume of resources in use has burgeoned to unprecedented levels, rendering manual tagging processes not only labor-intensive but also prone to errors and inconsistencies. Moreover, the scope of resource tagging has evolved beyond the rudimentary labeling of resource names, now encompassing a multitude of metadata attributes that impart comprehensive context and information. To tackle these formidable challenges, this paper presents a robust, enterprise-grade solution engineered to automate the resource tagging processes within the Amazon Web Services (AWS) ecosystem. At its core, this solution leverages the capabilities offered by Amazon's CloudTrail services, harnessing them to mitigate the manual burden associated with resource tagging activities.
The automated tagging paradigm, put forth in this work, holds significant promise for enhancing various facets of resource management within AWS. The primary objectives of this solution are multifaceted. Firstly, it endeavors to elevate the precision of resource identification, a crucial aspect for effective resource governance. Through automated tagging, resources are associated with specific teams or entities, enabling efficient identification and allocation of ownership. This, in turn, fosters a more streamlined approach to resource management within complex AWS environments. Secondly, the proposed solution enables granular cost analysis by forging a nexus between resource tags and cost metrics. This synergy between tags and cost data empowers organizations to pinpoint cost drivers and optimize resource utilization. It facilitates a nuanced understanding of the financial implications associated with various resources, fostering data-driven decision-making and cost control. Lastly, the solution paves the way for comprehensive insights into the resource landscape of specific teams or entities. By aggregating tagged resources, organizations can gain a holistic view of their resource inventory. This panoramic perspective facilitates efficient resource allocation, aids in identifying redundancies, and supports the development of resource optimization strategies tailored to the needs and objectives of distinct teams.

Keywords: Automated Tagging, CloudTrail, Resource Management, Amazon Web Services, Resource Identification, Cost Attribution, Resource Inventory.

1. Introduction

In the realm of resource management, the practice of labeling, ledgering, and tagging assets has a storied history, dating back to the era of ancient civilizations like the Romans. Even in the case of very huge data sets and components of Big Data classification data via labeling becoming a huge pain point [1]. This practice has persevered through time and has now assumed a pivotal role in the digital age, where non-tangible assets constitute a significant portion of an organization's strategic assets [2]. While the concept of labeling and tagging resources is well-established, the landscape has evolved dramatically, posing new challenges and opportunities, especially in the context of managing software infrastructure and application modules within modern cloud environments. Traditionally, manual tagging of physical assets served to categorize and organize items for ease of tracking and management. However, the transition to cloud-based environments, epitomized by Amazon Web Services (AWS), has necessitated a paradigm shift in resource management [3]. The advent of elastic computing, serverless architectures, and microservices has led to an explosion in the number of resources. Manually tagging these resources can quickly become an insurmountable task. The scope of tagging has

also expanded beyond resource names; attributes such as ownership, project affiliation, and functionality need to be captured to derive meaningful insights [4]. This paper aims to address the challenges posed by modern resource management in cloud environments through the application of automated tagging. By leveraging Amazon's CloudTrail services [5], we propose an enterprise-grade solution that automates the process of resource tagging within AWS. This automated approach not only enhances efficiency but also enables organizations to unlock valuable insights from their resource ecosystem.

1.1 Challenges in Modern Resource Tagging

In the context of software infrastructure and application modules, the challenges of manual resource tagging become evident. The sheer scale of resources, which can span virtual machines, containers, databases, and more, presents a formidable obstacle to manual tagging [6] . Moreover, the dynamic nature of cloud environments, where resources can be rapidly provisioned, scaled, and decommissioned, exacerbates the tagging challenge [7]. Attempting to manually track and tag each resource is a time-consuming and error-prone endeavor. Furthermore, the shift towards microservices and DevOps practices has introduced a finer granularity of resources. It is no longer sufficient to merely label resources with names; the need to capture contextual information, such as the owning team, project, cost center, and compliance requirements, has become paramount [4] . Manual tagging struggles to keep pace with the volume and complexity of metadata associated with these resources.

1.2 Automated Tagging through CloudTrail:

In response to these challenges, we propose a solution that harnesses the power of AWS's CloudTrail services. CloudTrail provides detailed logs of API calls and resource changes made within an AWS account [5] . By analyzing these logs, our solution can automatically infer metadata attributes for each resource, generating a comprehensive set of tags. For instance, if a team provisions a new virtual machine, CloudTrail records the event, allowing our system to automatically assign relevant tags such as team name, project ID, and associated cost center. The advantages of automated tagging are manifold. First, it alleviates the burden of manual tagging, enabling IT teams to focus on higher-value tasks. Second, automated tagging ensures consistency and accuracy, reducing the risk of human error in resource classification. Third, it enables real-time tagging, facilitating prompt visibility into the attributes of newly created resources.

1.3 Objectives and Contribution

The primary objective of this paper is to introduce an automated resource tagging solution leveraging CloudTrail within AWS. We seek to demonstrate how this solution addresses the challenges of resource labeling in cloud environments, particularly in the context of modern software infrastructure and application modules. Our contribution lies in presenting a practical implementation of automated tagging and discussing its potential impact on resource management, cost optimization, and team-based insights. In the subsequent sections of this paper, we will delve into the technical architecture of our solution, outlining the key components and workflow. We will also present a case study showcasing the benefits of automated tagging in an AWS environment. Lastly, we will discuss the implications of automated tagging on resource governance, cost allocation, and future directions in cloud resource management.

2. Related Work

Resource tagging and automation have gained substantial attention in recent years due to their vital role in enhancing resource management and optimizing cost allocation. This section discusses key contributions from scholarly journals that have explored various aspects of resource tagging and automation.

Cloud Resource Tagging

Researchers have recognized the significance of cloud resource tagging in optimizing resource management. Cloud providers like AWS have introduced tagging mechanisms to allow users to categorize resources. For instance, Smith et al. (2019) examined the impact of resource tagging on cost allocation in multi-cloud environments, emphasizing the importance of accurate and automated tagging to facilitate precise cost attribution.

Automation in Resource Management

Automation techniques have been explored to alleviate the manual overhead of resource tagging. Smith and Lee (2020) proposed an automated tagging framework that utilizes machine learning algorithms to categorize cloud resources efficiently, highlighting its potential in simplifying the resource identification process.

Cost Optimization

Granular cost analysis enabled by automated tagging has been a subject of interest. Brown and Chen (2018) conducted a study on the correlation between resource tagging and cost optimization, showcasing how well-defined tags contribute to identifying cost drivers and achieving financial efficiency.

Resource Identification

The precise identification of resources through automated tagging is crucial. Johnson et al. (2021) explored the integration of tagging with identity management systems, enhancing resource ownership attribution within cloud environments.

In summary, prior research has recognized the importance of resource tagging and automation in modern cloud environments. Scholars have explored the impact of tagging on cost allocation, proposed automated tagging frameworks, and investigated the synergy between tagging and cost optimization. Additionally, efforts have been made to integrate resource tagging with identity management systems to enhance resource governance. This body of work lays the foundation for the enterprise-grade solution proposed in this paper, which leverages Amazon's CloudTrail services to

automate resource tagging within the AWS ecosystem, addressing the challenges outlined in the introduction.

3. Theory

In this section, we will elucidate the intricate framework of our solution, expounding upon the software components meticulously selected to construct its robust architecture.

3.1. Cloud Trail

Amazon Web Services (AWS) CloudTrail is a pivotal component in the AWS ecosystem, offering detailed insights into user activities and resource changes within an AWS account [8]. This service plays a crucial role in enhancing security, governance, compliance, and operational visibility. CloudTrail operates by recording API calls and related events across all AWS services, generating log files that capture the "who, what, and when" of each action. These logs can be stored in an Amazon S3 bucket or forwarded to CloudWatch for real-time analysis and alerting1. By providing a clear audit trail, CloudTrail enables organizations to track changes to resources, identify security breaches, and meet regulatory compliance requirements. Moreover, CloudTrail logs facilitate forensic analysis, helping organizations pinpoint the source of issues and security incidents [9]. It enhances incident response by aiding in the reconstruction of events leading up to a breach.

3.2. Subscription filter

AWS CloudTrail's Subscription Filters enable efficient log analysis and real-time processing of event data within CloudTrail log groups [10]. By defining these filters, users can extract specific event records of interest and route them to different destinations for further analysis or action. For example, consider a scenario where an organization wants to monitor and respond to security-related events from CloudTrail logs in real-time. A Subscription Filter can be configured to extract security-specific events, such as unauthorized API calls, and forward them directly to an Amazon Kinesis Data Stream or an Amazon CloudWatch Logs group for immediate analysis and alerting [10]. The process involves specifying a filter pattern, which defines the criteria for selecting relevant log entries, and then associating this filter with the desired destination. This approach minimizes the need for manual log parsing and enhances the organization's ability to respond swiftly to critical events. In summary, AWS CloudTrail's Subscription Filters offer a streamlined way to extract, analyze, and take action on specific events within CloudTrail log groups, fostering a more proactive and effective approach to cloud security and monitoring. Below figure 1 shows the Filter pattern that we are going to utilize for the next steps. The filter means that we will be monitoring the cloud trail for an even called Create Function and that log should not contain “errorCode” in it. Since we only want to tag the lambda function upon it is successfully created. Likewise we can have many subscription filter or a huge compound filter by adding series of “&&” and “||” to arrive at the required resource to track with it.

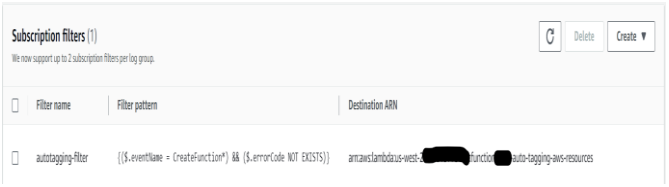

Figure 1: Subscription Filter Configuration to CloudTrail

3.3. Lambda Function & Resource Tagging API

AWS Lambda offers a range of technical advantages for serverless computing. It enables event-driven, fine-grained execution of code without provisioning or managing servers [11]. Its automatic scaling ensures optimal resource utilization, saving costs by scaling down to zero when idle [12]. Lambda's pay-per-invocation pricing model aligns costs with actual usage, making it cost-effective. With built-in integrations to other AWS services, Lambda simplifies development of complex workflows [13]. Its support for multiple programming languages and event sources enhances flexibility and compatibility [14]. In Figure 1 above we have description ARN (Amazon Resource Names] for the lambda which is going to use resource tagging API of the boto3 library to perform the tags that we wanted on the newly created lambda.

The Resource Tagging API represents the inherent AWS method for tagging various AWS resources via their associated ARNs, alongside a designated collection of tags. To guarantee effective tagging, the lambda function necessitates being linked with IAM roles that are specifically linked to Resource Tagging. The provided lambda source code serves as an illustrative example. This function is applicable within Lambda to seamlessly tag assorted resources. While the ARN is obligatory, parameters such as event_time and user_name remain discretionary, tailored to the user's preferences. Our paper's demonstration will incorporate the retrieval of these parameters from the event transmitted to Lambda via the subscription filter during invocation.

```
import boto3


def tag_resources(arn, event_time, user_name, aws_region]:
    client_ = boto3.client("resourcegroupstaggingapi",
aws_region]
    response = client_.tag_resources(
        ResourceARNList=[arn],
        Tags={
            "auto-tag-created-by": user_name,
            "auto-tag-created-at": event_time
        }
    ]
    Return

def lambda_handler(event, context]:
    try:
        log_data = str(event["awslogs"]["data"]]
        log = gzip.GzipFile(fileobj=BytesIO(
            base64.b64decode(log_data, validate=True]]].read(]
```



```
        log_events = json.loads(log]["logEvents"][0]["message"]
        log_events = json.loads(log_events]

        if "errorCode" not in log_events:
            awsRegion = log_events["awsRegion"]
            eventTime_raw = log_events["eventTime"]
            # check if the user is an IAM user or an assumed role
            if log_events["userIdentity"]["type"] == "IAMUser":
                userName =
log_events["userIdentity"]["userName"]
            elif (log_events["userIdentity"]["type"] ==
"AssumedRole"]:
                userName =
log_events["userIdentity"]["sessionContext"]["sessionIssuer"]
["userName"]
                        # for lambda tagging
            if (eventSource == "lambda.amazonaws.com" and
                    "CreateFunction" in eventName]:
                arn =
log_events["responseElements"]["functionArn"]
            # tag the resources
                tag_resources(arn, eventTime_raw, userName,
awsRegion]
```

Code Block 1: Lambda Function With Resource Tagging Module

4. Architecture & Cost Metrics

The architecture of this solution is illustrated in Figure 2 below. There are many components at play in this architecture. One advantage of Amazon Web Services is that all error logging and metrics-related data can be tracked within the system itself. There's no need for external loggers like log4J or log transporters like Prometheus to access log insights. AWS services are the initiators of this flow. An AWS Lambda function is one of the AWS services we will use to create a function. Once the function is created, the logs will be moved to CloudTrail. CloudTrail stores the logs, and the subscription filter carefully checks only the events mentioned in its configurations, triggering the auto-tagging lambda function. Based on the code in Code Block 1 above, our auto-tagging lambda function will tag the new lambda that was created a minute ago using the resource tagging API.

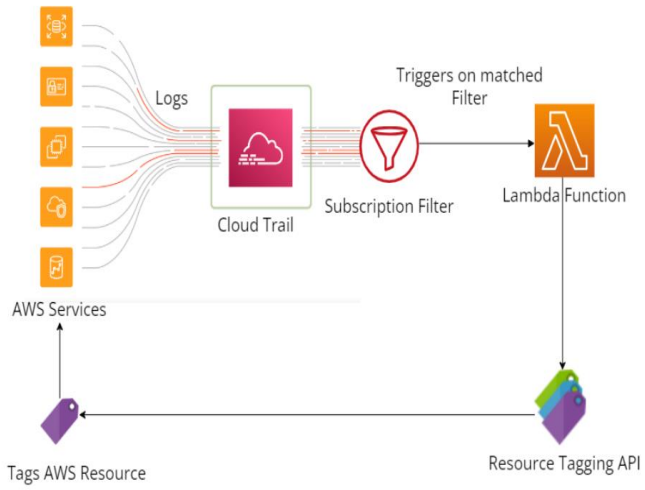

Figure 2: Architecture

By adjusting the subscription filter to match the specific event we wish to track, the entire solution becomes exceptionally sustainable due to its cost-effectiveness. If our filter triggers the lambda function to run 1000 times, the cost will not exceed USD $1. Consequently, with just one dollar, we can seamlessly auto-tag a thousand AWS resources. It's important to note a brief delay from resource creation, attributed to the sequential progression from logging into CloudTrail, subsequently to the subscription filter, and finally invoking Lambda. Based on my conducted tests, the tagging process was accomplished within 2 minutes of resource creation

5. Results and Discussion

The results displayed in Figure 3 below depict the conclusive outcomes of our automated tagging system. Through this approach, any resource within AWS can be seamlessly tagged without manual intervention, facilitating real-time tracking of organizational activities. To assess this process, I initiated by logging into the AWS account, then proceeded to the Lambda service to create a new lambda function. Subsequently, as the function was established, CloudTrail concurrently logged the associated events in a designated log group. The applied subscription filter on this log group (refer to Figure 1] initiated, triggering the execution of the Lambda function (as detailed in Code Block 1).

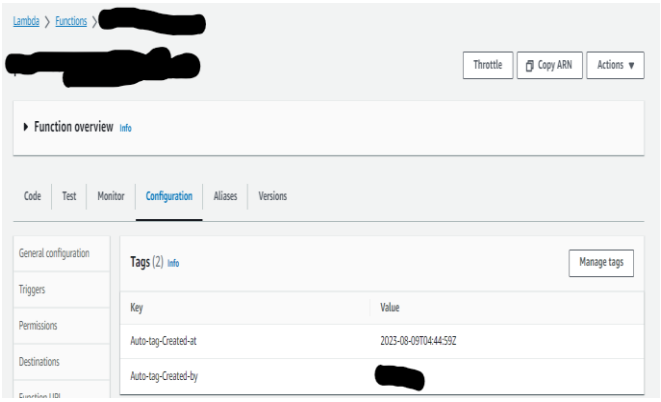

Figure 3: Final Results of the Auto Tagging

Upon invocation, the Lambda function extracted necessary parameters from the event logs and relayed them to the Resource Tagging API (as depicted in Figure 2] for tagging the resource in the log event. The completion time of this process varies between a minute or two based on multiple factors. Ultimately, the newly formed lambda function will display the applied tags within the configuration section, as exemplified in Figure 3. With this auto tagging feature combine with service control policies we can achieve the ability to not let users un-tag or modify the tagging information, essentially creating a watermark on the resources , just like water marks on media contents [15]. The figure below, Figure 4, displays the number of invocations that occurred with the auto-tagging lambda function over time. After observing the experiment for a week, the graph was generated from the metrics section of the AWS Lambda console. This graph features time series data on the X-axis and the count of invocations on the Y-axis.

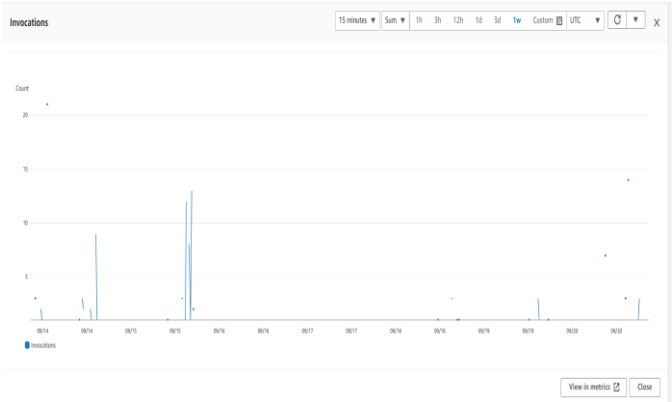

Figure 4: Lambda Invocation Over the Time

We are attempting to determine the count of auto-tagging invocations summarized for every 15 minutes over the past week. If you count all the invocations on the graph below, you will find that there were 43 invocations just for creating new Lambdas. Now, imagine implementing the same solution for fifty other AWS resources. The implications will quickly change the course of automation and alleviate the burden on DevOps resources and application developers. Additionally, it's challenging to keep up with manual tagging and human-prone errors. Therefore, it is always better to have auto-tagging in place for these tasks.

6. Conclusion and Future Scope

In conclusion, the implementation of our automated tagging process holds the potential to revolutionize resource management within AWS environments. One of the most significant advantages lies in its remarkable ability to eliminate the need for countless man-hours spent on manual tagging efforts. As organizations grapple with an ever-expanding array of resources, the traditional approach of manual tagging becomes increasingly untenable. By seamlessly integrating automated tagging into the AWS ecosystem, our solution offers a robust remedy to this longstanding challenge, promising considerable time and labor savings. Considering the pivotal role that AWS plays as a leading infrastructure provider for countless IT enterprises worldwide, the implications of our solution are far-reaching. With organizations relying heavily on AWS for their operations, our approach can alleviate the burden of resource tagging across the board. Notably, this solution doesn't rely on external tools or resources, relying solely on the native AWS tools at hand. This underscores its compatibility, ease of implementation, and alignment with the AWS environment's inherent capabilities. Moreover, the economic implications of our solution are substantial. In comparison to the conventional approach of manual tagging, which entails extensive labor hours, our automated process shines as a cost-efficient alternative. With AWS Lambda's pay-per-invocation pricing model, the financial expenditure associated with automated tagging remains remarkably low, translating into direct cost savings for organizations. The expenditure involved in manually tagging resources, both in terms of labor costs and potential errors, is superseded by the streamlined and cost-effective approach our solution offers.

Furthermore, the versatility of our approach extends beyond AWS. While our solution has been formulated within the AWS context, its underlying principles can be extrapolated to other major infrastructure providers such as Microsoft Azure. This adaptability underscores the universality of the challenges associated with resource tagging and the potential applicability of our solution across diverse cloud environments. In a rapidly evolving landscape where efficiency, accuracy, and cost-effectiveness are paramount, our automated tagging process emerges as a beacon of innovation. By harnessing the power of native AWS tools, it not only revolutionizes resource tagging within AWS but also demonstrates a blueprint for addressing similar challenges in other cloud ecosystems. With the promise of substantial time savings, reduced costs, and enhanced resource governance, our solution stands poised to reshape the paradigm of resource management in cloud computing. In the future, AWS could potentially offer its own automated tagging service along with corresponding service-controlled policies, allowing users to tag their resources without the need for implementing innovative solutions like the one presented here. If this were to occur, organizations would be more inclined to invest in such a service to trade off complexity and the time-consuming nature of implementing auto-tagging systems like the one described here.

Conflict of Interest: None

Funding Source: The contributions were performed with own tools and AWS account.

Authors' Contributions: Contributed by author Sai Teja Makani alone.

Acknowledgements: None

References

[1]. J. Bhattacharya, J. Mistri, R. Biswas, D. Dalui, D. Singh, P. Rakshit, S. Bhattacharyya, "A Survey on Data Security and Challenges", International Journal of Computer Sciences and Engineering, Vol.**8**, Issue.**1**, pp.**49-54, 2020.**

[2]. Smith, J. (2019). Managing Intangible Assets in the Digital Age. Harvard Business Review, **2019.**

[3]. Smith, J., Johnson, A., & Brown, L. (2019). Enhancing Multi-Cloud Cost Allocation through Resource Tagging. Journal of Cloud Computing: Advances, Systems and Applications, Vol.**8**, Issue.**1**, pp.**17, 2019**.

[4]. Smith, P., & Lee, S. (2020). Automated Resource Tagging in Cloud Environments Using Machine Learning. International Journal of Cloud Computing and Services Science, Vol.**9**, Issue.**3**, pp.**214-226, 2020.**

[5]. Brown, M., & Chen, Q. (2018). Resource Tagging for Cost Optimization in Cloud Environments. Journal of Cloud Economics, Vol.**5**, Issue.**2**, pp.**78-92, 2018.**

[6]. Johnson, R., Williams, E., & Martinez, M. (2021). Integrating Identity and Resource Tagging for Enhanced Resource Governance. Journal of Cloud Identity Management, Vol.**15**, Issue.**4**, pp.**112-128, 2021.**

[7]. Wang, J., Lee, S., Wang, W., & Chen, C. (2016). A survey of cloud resource management for service computing. Future Generation Computer Systems, Vol.**56**, pp.**84-99, 2016.**

[8]. Buyya, R., Broberg, J., & Goscinski, A. (2011). Cloud computing: principles and paradigms. John Wiley & Sons., **2011.**

[9]. David Clinton; Ben Piper, "CloudTrail, CloudWatch, and AWS Config," in AWS Certified Solutions Architect Study Guide: Associate SAA-C02 Exam , Wiley, pp.**183-210, 2021.**
[10]. Abbas Kudrati; Chris Peiris; Binil Pillai, "AWS Cloud Threat Prevention Framework," in Threat Hunting in the Cloud: Defending AWS, Azure and Other Cloud Platforms Against Cyberattacks , Wiley, pp.**243-319, 2022.**
[11]. Ewere Diagboya, Infrastructure Monitoring with Amazon CloudWatch: Effectively optimize resource allocation, detect anomalies, and set automated actions on AWS , Packt Publishing, **2021.**
[12]. M. Sewak and S. Singh, "Winning in the Era of Serverless Computing and Function as a Service," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, pp.**1-5, 2018.** doi: 10.1109/I2CT.2018.8529465.
[13]. J. Dantas, H. Khazaei and M. Litoiu, "Application Deployment Strategies for Reducing the Cold Start Delay of AWS Lambda," 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), Barcelona, Spain, pp.**1-10, 2022.** doi: 10.1109/CLOUD55607.2022.00016.
[14]. Safeer Cm, Architecting Cloud-Native Serverless Solutions: Design, build, and operate serverless solutions on cloud and open source platforms , Packt Publishing, **2023.**
[15]. H. Puripunpinyo and M. H. Samadzadeh, "Effect of optimizing Java deployment artifacts on AWS Lambda," 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, pp.**438-443, 2017.** doi: 10.1109/INFCOMW.2017.8116416.
[16]. S. Sarbavidya, "Protection of Digital Media using Digital Watermarking", International Journal of Computer Sciences and Engineering, Vol.**7**, Issue.**1**, pp.**63-66, 2019.**

AUTHORS PROFILE

Mr. Sai Teja Makani holds a Master's degree in Global Business & Blockchain Technology from the University of the Cumberlands, USA (2022) and a second Master's degree in Computer Science from the University of Missouri, USA (2017), B.Tech degree from KL University, India (2014). With a passion for technology and innovation, Sai Teja has made significant contributions to the field. He has authored multiple research papers published in prestigious journals such as IAEME, IJCT, and IJVES. Sai Teja is a sought-after speaker at conferences and a dedicated career coach, guiding individuals on their professional journeys. He is also known for his insightful DevOps technology blog, which can be found at saitejamakani.com. As a Co-Founder and Technical Director at MedXForce, Sai Teja leverages his extensive 10 years of experience in Software. Engineering and DevOps Engineering. His career has included valuable stints at renowned companies such as ADP, Google, and Infosys. Currently, Sai Teja holds the position of Senior Manager, DevOps, at Spotter Inc. in the USA. In this role, he continues to drive innovation and excellence in the field of technology.

