ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14885967

Automated Cleanup of Unused AWS Cloud Formation Resources Using AWS Resource Tags and Lambda Functions

Prudhveer Reddy Kankar

DevOps Engineer

Publication Date: 2025/02/18

Abstract: The AWS cloud platform has experienced rapid growth due to its expanding features and support for on-demand access to compute, storage, networking, and virtualization. AWS CloudFormation is a service that enables developers and busi- nesses to create, provision, and manage a collection of related AWS and third-party resources in an orderly and predictable manner. AWS CodePipeline can be utilized to deploy AWS CloudFormation stacks, enhancing continuous integration and continuous delivery (CI/CD) capabilities. Companies often use multiple AWS accounts for different environments and deploy resources across them. As part of the CI/CD process, a central AWS account is used to deploy CloudFormation stacks to other accounts using AWS CodePipeline. However, when an application is no longer needed, there is no straightforward way to use the centralized account to delete the unused CloudFormation resources. While CloudFormation stacks can be updated and modified from the central account using AWS CodePipeline, deleting them remains a challenge. This paper discusses an approach to address this issue using CloudFormation tags and AWS Lambda.

Keywords: *AWS Code Pipeline, AWS Lambda, CI/CD, Cloud Resource Management, Tags, Automation, AWS Cleanup Processes, Software Development Processes.*

How to Cite: Prudhveer Reddy Kankar. (2025). Automated Cleanup of Unused AWS Cloud Formation Resources Using AWS Resource Tags and Lambda Functions. *International Journal of Innovative Science and Research Technology*, 10(1), 2563-2564. https://doi.org/10.5281/zenodo.14885967.

I. INTRODUCTION

AWS CodePipeline is a native AWS service widely used in CI/CD processes. It allows the deployment of AWS Cloud- Formation stacks across multiple AWS accounts from a single centralized account. Companies typically use different AWS accounts for various environments, such as development, infrastructure, testing, and production. AWS CodePipeline can be connected to a source, such as GitHub, containing the AWS CloudFormation template and its associated resources. Whenever a change is made to the source, AWS Code- Pipeline automatically updates the CloudFormation resources in multiple accounts. In real-life production scenarios, these stacks can deploy a significant number of resources, such as ECS, VPC, and S3, which are cost-intensive. While creating and updating CloudFormation resources using CodePipeline is straightforward, deleting the resources when they are no longer needed is not. This paper presents an approach to delete CloudFormation stacks using CloudFormation tags.

According to [1], tagging AWS resources is considered one of the best practices for efficient resource management, which aligns closely with the automated cleanup process described in this paper.

II. METHODOLOGY

CloudFormation templates are JSON or YAML files that define all the resources required by an application. AWS resources can be tagged using custom key-value pairs. This method suggests using CloudFormation tags to identify and delete deployed stacks. A custom tag with the application name should be added to the CloudFormation stack template. A Lambda function needs to be created in the central account with the following functionality: the function should connect to the CloudFormation client in all the accounts where the CloudFormation resources have been deployed using the STS assume role functionality. It should be able to take the applica- tion name as input, search for all the CloudFormation stacks in the AWS accounts using boto3 that have that particular application name as a tag, identify the specific stack, and delete it as necessary.

In [2], researchers discuss the integration of AWS Lambda with CI/CD pipelines for automating resource management tasks, which further validates the proposed methodology.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14885967

III. IMPLEMENTATION

Whenever an application is no longer needed, the application name can be passed to the Lambda function, which will delete all the stacks in all the accounts. This approach helps reduce the cost of unused resources, especially when an application is deployed to a large number of accounts, eliminating the need to manually delete the stacks.

In my experience, we faced a similar challenge of managing and deleting unused AWS CloudFormation resources across multiple AWS accounts. We implemented the proposed solu- tion using CloudFormation tags and AWS Lambda. Here is a step-by-step overview of our implementation:

- Tagging CloudFormation Stacks: We added a custom tag with the application name to each CloudFormation stack template deployed across various AWS accounts.
- Creating the Lambda Function: We developed a Lambda function in the central AWS account. This function connects to the CloudFormation client in all the accounts using the STS assumed role functionality.
- Searching and Deleting Stacks: The Lambda function takes the application name as input, searches for all the CloudFormation stacks with that tag using boto3, identifies the specific stacks, and deletes them as necessary.
- Automating the Process: We automated the process by integrating the Lambda function with our CI/CD pipeline. Whenever an application is no longer needed, the application name is passed to the Lambda function, which then deletes all the related stacks in all the accounts.

This implementation has significantly reduced the cost of unused resources and improved our resource management efficiency. Research findings in [3] emphasize the importance of automation in reducing operational costs in cloud environments, which aligns with our results.

IV. POTENTIAL CHALLENGES

➢ *Implementing this Solution Comes with its Own Set of Challenges:*

- Rate Limiting and Throttling: AWS services have rate limits that can cause throttling issues when making a large number of API calls in a short period. This can be mitigated by implementing exponential backoff and retry mechanisms [4].
- Cross-Account Permissions: Ensuring that the Lambda function has the necessary permissions to assume roles and perform actions across multiple AWS accounts can be com- plex. Proper IAM role configurations and trust relationships are essential.
- Error Handling and Logging: Robust error handling and logging mechanisms are crucial to identify and troubleshoot issues during the deletion process. This includes handling partial deletions and ensuring idempotency.
- Resource Dependencies: Some resources may have de- pendencies that need to be carefully managed during deletion to avoid breaking other services. Using the DependsOn at- tribute in CloudFormation can help manage these dependencies.
- Security Considerations: Ensuring that the Lambda function and associated resources are secure and follow best practices for AWS security is vital. This includes encrypting sensitive data and using secure communication channels [5].

V. RELATED WORK

Several studies and implementations have explored the use of AWS CloudFormation, AWS CodePipeline, and AWS Lambda for managing cloud resources. These resources pro- vide valuable insights into the capabilities and best practices for using AWS services in CI/CD processes. For instance, [6] discusses best practices for managing AWS resources in a multi-account environment, which supports the techniques detailed in this paper.

VI. CONCLUSION

This paper presents a method to clean up unused AWS CloudFormation resources in AWS accounts using CloudFormation tags and AWS Lambda. By leveraging custom tags and a centralized Lambda function, companies can efficiently manage and delete unused resources, thereby reducing costs and improving resource management.

REFERENCES

[1]. AWS Documentation. "Tagging AWS Resources." Available: https://docs.aws.amazon.com/general/latest/gr/aws_tagging.html, Accessed: Jan. 23, 2021.
[2]. S. S. Gill, I. Chana, M. Singh, and R. Buyya, "Efficient Management and Allocation of Resources in Serverless," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1006-1019, 2019.
[3]. S. S. Gill, I. Chana, and R. Buyya, "Modeling and Optimization of Performance and Cost of Serverless Computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 964-977, 2021.
[4]. AWS Documentation. "Error Retries and Exponential Backoff in AWS."Available:https://docs.aws.amazon.com/general/latest/gr/api-retries.html, Accessed: Jan. 23, 2021.
[5]. J. Li, L. Yu, J. Zhang, and Z. Li, "A Survey of Security in Cloud Computing," *IEEE Access*, vol. 6, pp. 64724-64736, 2019.
[6]. M. Ali, A. R. Butt, and M. F. Younis, "Resource Management and Allocation in Multi-Cloud Environments: A Survey," *IEEE Access*, vol. 8, pp. 23524-23542, 2020.