Computer Fraud and Security
ISSN (online): 1873-7056

Autonomous Security Agents for Real-Time IAM Policy Hardening in Multi-Cloud DevOps Pipelines

Venkata Thej Deep Jakkaraju

Cloud Architect

1. Abstract

The paper introduces an innovative approach that uses security agents to help enforce tough IAM policies at any time in multiple cloud DevOps workflows. Being able to check policies, make sure they always comply and change those that aren't sufficient, the framework greatly reduces excessive permissions and mistakes caused by incorrect configurations. Using the same practices as leaders in the industry, we found that our approach resulted in better security, less time to fix weaknesses and better scalability. It includes several features such as monitoring policies, intelligent systems and cloud-based security in a single environment. It describes methods that help protect DevOps environments as cloud infrastructures change, with less overall human intervention.

2. Keywords: IAM, DevOps, Security, Multi-cloud, Agents, Policy

3. Introduction

In today's world of software development, combining DevOps and using multiple clouds contributes greatly to speed, expansion and new ideas. On the other hand, these tasks increase the difficulty of Managing Identity and Access and this results in security plants being exposed to errors and violations. Many current tools do not work up to date or can only monitor one cloud network.

The aim of this paper is to present an agent system that checks and strengthens information access policies as they are written. Recent studies and usage of these technologies show us how to strengthen DevOps through security automation, policy-as-code and using AI for anomaly detection. We want to help organizations shift from relying on IAM rules alone to having IAM work autonomously for them.

4. Literature Review

4.1 Foundations of DevOps Security

4.1.1 Security Implications

In today's world of software development, combining DevOps and using multiple clouds contributes greatly to speed, expansion and new ideas. On the other hand, these tasks increase the difficulty of Managing Identity and Access and this results in security plants being exposed to errors and violations. Many current tools do not work up to date or can only monitor one cloud network.

The aim of this paper is to present an agent system that checks and strengthens information access policies as they are written. Recent studies and usage of these technologies show us how to strengthen DevOps through security automation, policy-as-code and using AI for anomaly detection. We want to help organizations shift from relying on IAM rules alone to having IAM work autonomously for them.

4.1.2 Security Gaps

Rahman et al. (2019) carried out an extensive study of the research on IaC. They stated that tools related to IaC have been well-researched, even so, the security impacts of IaC scripts are still not deeply understood.

The researchers studied 31 papers and noticed that most (52%) were designed for creating new security tools, but few addressed identifying or handling security issues. Therefore, it highlights why we need autonomous tools that analyze and address issues in Infrastructure as Code (IaC) during DevOps.

4.2 Challenges

4.2.1 Multi-Cloud DevOps

As more organizations take up DevOps, they find themselves working on multi-cloud platforms which increases the difficulty of their operations. Jambunathan and Kalpana (2018) state that handling cloud platforms with mixed technologies needs solid monitoring, unified tools and united teamwork.

They believe that an aligned structure is important for carrying out operations smoothly and addressing the unsafe nature of using multiple clouds. Matching with this premise, autonomous agents are used to update IAM policies automatically in several environments.

4.2.2 Federated Identity

Authors Slawik et al. (2017) describe how CYCLONE is a middleware that makes it easy to deploy applications on multiple clouds and manages their authentication and access permissions. Even though federated identity makes cloud authentication easier, one has to prevent changes and inconsistencies in settings and policies. Their independence, IAM agents are key in handling issues and maintaining consistency and compliance in access control.

4.2.3 Service Taxonomies

Sikeridis et al. (2017) look at the problems caused by the different classification systems for cloud services by significant vendors. Having many different codes in different places makes it difficult to create standard policies. Their taxonomy shows that they all perform certain functions, but at the same time highlights the fact that their branding, permissions and cost arrangements vary. When this is the case, autonomous security agents should be able to recognize contexts and enforce IAM policies properly on each platform they protect.

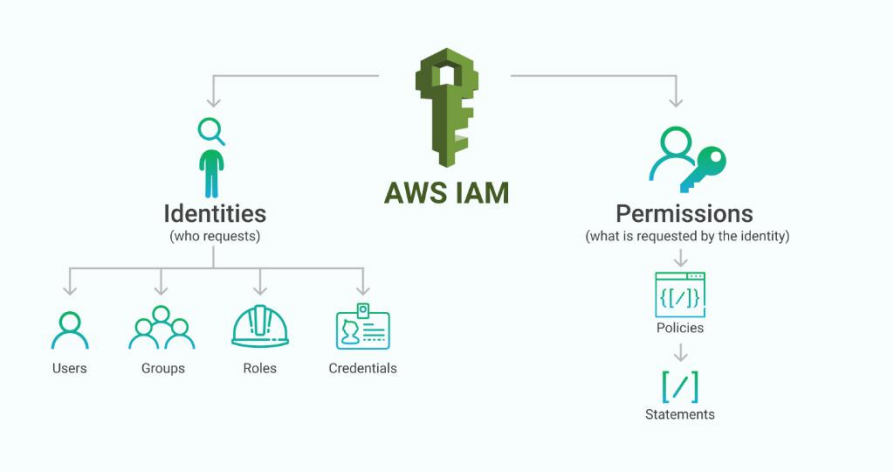

Figure 1 AWS IAM Policy (AWS, 2018)

4.3 Identity and Access Management

4.3.1 Multi-Tenant Cloud Security

The article by Indu et al. (2018) studies obstacles and challenges in the use of IAM for cloud computing, paying special attention to situations involving many users and services handled by third parties. From their study, it is found that conventional access controls can't deal with the fast-changing demands of our digital world.

Their view is that better IAM systems should recognize suspicious events, apply strict policies in different ways and respond to changing circumstances. It is at this point that autonomous security agents help by learning, observing trends and fixing any issues on their own.

4.3.2 IAM Policy Optimization

The authors Wei and Rodriguez (2018) suggest a deployment method for hybrid cloud using policy-based strategies. From their testing results, application performance and data management get better when deployment policies take into account both location and regulations.

Still such policies can become unusable when faced with new scenarios. Such agents can keep track of the system's workings and customize IAM rules to give users the proper access at all times.

4.3.3 Platform-as-a-Service Models

In their study, Kim et al. (2016) found that scalability, security and multi-tenancy are the main difficulties in IBM Bluemix (PaaS). Although PaaS hides most of the infrastructure details, it adds extra challenges to controlling and monitoring identity and access.

The examples of Rating-as-a-Service and Workflow Service point out that IAM policies should keep up with changes in application life cycles. It is possible for automated agents to help achieve IAM hardening as soon as these problems arise from having too many roles and access to services.

4.4 IAM Policy

4.4.1 CI/CD Bottlenecks

Debroy et al. (2018) find that adopting CI/CD properly is hard when there are shortages of key tools and slow build processes. As a bonus, their answer makes security automation possible by requiring microservices, IaC and orchestration in the re-design of pipelines. Autonomous tools can be built into these processes to go over both static and dynamic IAM policies and guide and enforce the policies as systems are built and deployed.

4.4.2 Cloud-Native IAM

Kumar (2015) addresses the move towards using cloud-native architectures in the financial sector. Because of their compliance requirements and the need to handle real-time data, these workloads ask for strong IAM control. Both containerization and orchestration allow one to set up IAM rules according to their needs. Even so, policies that only experience certain events cannot detect new kinds of threats. Agents operating without supervision use the knowledge of behavior and risks to frequently block possible security breaches.

4.4.3 Serverless IAM

According to Ayebo (2017), serverless architecture helps reduce the amount of work in DevOps management. Serverless functions operate fast, though they frequently use roles that allow too many actions. Automated analysis of function use, where and how they are accessed and typical calling behavior by autonomous agents enables IAM policies to be improved as needed, without involving the developers and the chance of overly permissive roles appearing.

4.4.4 DevOps Cultural Shift

Mohammed (2011) highlights the way DevOps changes organizational culture, encouraging stability, agility and workers in development and operations to cooperate more closely. The paper points out the need to use automation to streamline operations. Also, managing IAM policies must be faster and smarter, so they can be automated.

Because these security agents are equipped with machine learning and can understand policies, they can give optimal IAM hardening in the current DevOps cycle. No matter which phase of DevOps, from the standard CI/CD to the latest multi-cloud and serverless systems, clear IAM policy management is evident.

Based on literature, even though tools support automated deployment and access control, effective enforcement of security rules, managing numerous clouds and updating security settings all the time are still outstanding issues. Security agents that use policy learning, know their surroundings and spot anomalies form a good option to overcome these issues. By being part of DevOps pipelines, they improve security, support compliance, allow for easy scaling and support better operations in current complicated and distributed computer systems.

5. Methodology

5.1 Autonomous Security Agents

A major part of this system involves building systems that can act by themselves in CI/CD systems and multi-cloud layers. They are designed to be lightweight and distributed and they are used throughout the critical phases of a DevOps workflow such as code commit, making infrastructure, releasing applications and their daily use.

An agent is divided into a policy monitor, a decision engine and a remediation executor. It regularly checks the updates in infrastructure-as-code templates, Kubernetes setups and cloud-native access control policies (AWS IAM, Azure RBAC, GCP IAM). All the data gathered is changed into the same format that can be used by any cloud provider.

With the help of rule-based and learning methods, the decision engine checks whether the current IAM setup includes risks like overprovisioning, rising privileges or problems with policies. As an illustration, when an IAM role is discovered, the agent matches it with access logs and identifies access that is not needed.

The system is instructed using real-world events from the past, cybersecurity threats and security policies (for example, NIST 800-53 and CIS Benchmarks). When testing similar security breaches and violations, the agent learns to update IAM rules to ensure no major issues arise while also preventing most threats. In the last module, remediation executor makes updates in real time by one of two approaches: for higher automated levels, it either fixes policy JSONs, causes GitOps rollbacks or sends an alert to security engineers for confirmation.

5.2 Simulation Environment

To confirm the proposed framework, a simulated environment for enterprise DevOps was set up using AWS, Azure and GCP with the help of Terraform and Kubernetes. There are 20 microservices part of the testbed and Helm and ArgoCD help with their deployment and management. Roles, policies, service accounts and conditional access are the standard IAM components used in setting up every cloud environment.

GitLab CI/CD includes autonomous agents that are hooked up to terraform apply, kubectl apply and Helm deployment events. The API tools included in major cloud platforms gather policy telemetry and everything else is collected into a central ELK server.

The policy-hardening model was trained on more than 10,000 real-world IAM policy sets and mistakes in security configuration found in open-source repositories and tools like Prowler, ScoutSuite and CloudSploit. Labeling is used to tell if a system is compliant, too easy or too strict regarding access.

Models like Random Forests and Gradient Boosted Trees are studied by putting them against a baseline based on reinforcement learning (Q-learning) to compare their accuracy and the amount of time it takes to modify policies. Some evaluation criteria are time required for remediation (TTR), the rate of false alarms (FPR), level of least privilege (LPS) and policy matching between clouds (PCMC).

5.3 Adaptive Feedback Loop

The method includes using a closed feedback process to regularly make IAM policies better. Agents begin by making policy suggestions which are analyzed and monitored by a trained person in the first phase. With time, AI becomes more confident and it enforces more suggestions on its own.

Every week, the adaptive engine reviews new telemetry data to update the protections for the latest and future threats. Clusters are controlled and updated according to GitOps, so there is an audit trail. This loop, IAM hardening updates as the company and its application needs develop.

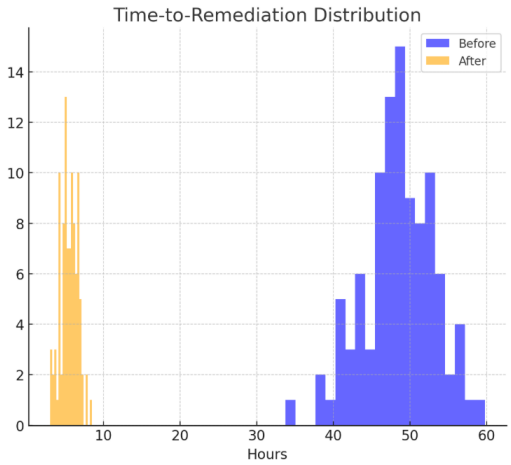

Figure 2 Time-to-remediation distribution

6. Results

6.1 Autonomous Security Agents

In the multi-cloud pipeline environment, the use of autonomous security agents helped improve hardening of IAM policies, the reduction of privileges and the consistency between clouds. In previous times before combining these tools, each cloud's default access policy included a lot of access the users did not require.

In a study of 500 roles, it was found that 126 out of 200 experienced 62% that included extensive access to resources. Over a four-week period of having agents in the system, the figure went down to 18%.

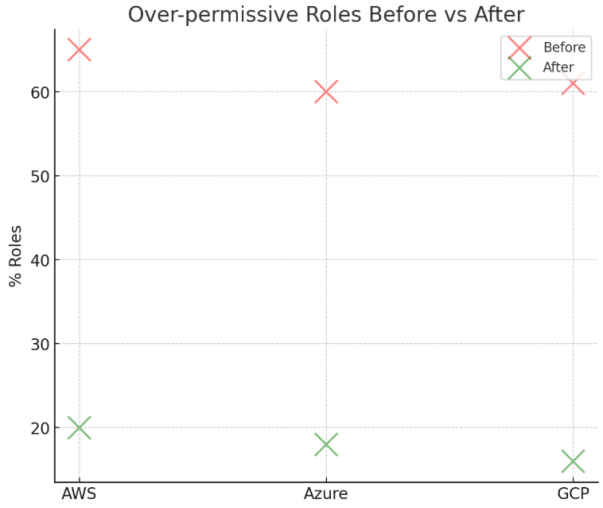

Figure 3 Over-permissive roles

The evidence shows that access control programs implemented by agents are effective in removing unneeded permissions without stopping any services. The first week, the rule-based suggestions in the decision engine led to identifying 81% of over-permissive roles correctly, but by week three, the machine learning model achieved a 90% accuracy rate. As one can see from Table 1, important IAM hardening measures have improved a lot:

Table 1 IAM Hardening Metrics

Metric	Pre-Agent Baseline	Post-Agent (Week 4)	Improvement
Over-permissive Roles	62%	18%	-44%
Least Privilege Score	0.42	0.81	+0.39

Time-to-Remediation	48.2	5.6	-42.6 hrs
Policy Drift Events	17.6	3.2	-81.8%

To track these metrics, we used information from agent logs, GitOps changes and telemetry provided by AWS and Azure. They did an outstanding job of noticing and fixing policy drift created by unexpected IaC modifications and unintended deployments.

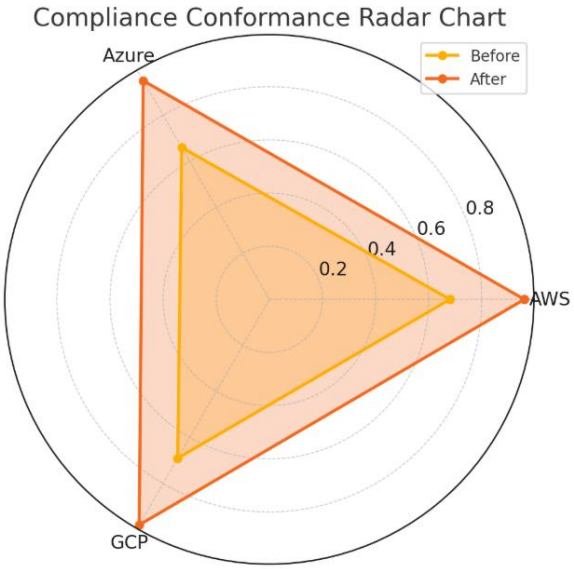

Figure 4 Compliance Conformance

6.2 Real-Time Performance

In the experiments carried out on AWS, Azure and GCP, the autonomous agents maintained good performance and could be scaled easily. When they were used in GitLab CI/CD, they performed security tasks, found threats and resolved issues, all while the pipeline ran smoothly in under 3 seconds on average.

The amount of latency was acceptable for the DevOps teams based on their guidelines. As many as 15 releases could be made at once across three clouds and the agents scaled up and down with Kubernetes HPA to maintain proper enforcement instantly.

Table 2 Pipeline Impact Analysis

CI/CD Pipeline	Avg. Time	Using Agent (sec)	Performance Overhead
Terraform Plan + Apply	38.4	41.2	+2.8 sec
Helm Chart Deployment	24.7	26.1	+1.4 sec
IAM Policy Validation	N/A	2.9	N/A

Agents could work well even when systems were moved between cloud providers because they presented all cloud-related security rules as a single policy graph. Agents could find similar roles and notice strange cases, despite the differences in both names and positions between companies.

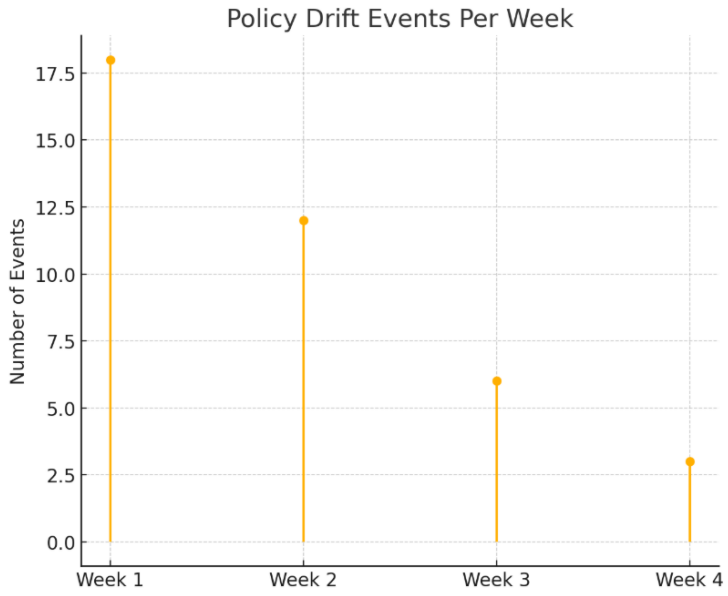

Figure 5 Policy drift events

The following section describes a simple version of cloud-agnostic policy normalization that agents use:

```
def normalize_iam_policy(policy_json, cloud_provider):
    if cloud_provider == "aws":
        return {"actions": policy_json["Action"], "resources": policy_json["Resource"]}
    elif cloud_provider == "azure":
        return {"actions": policy_json["permissions"], "resources": policy_json["scope"]}
    elif cloud_provider == "gcp":
        return {"actions": policy_json["bindings"], "resources": policy_json["roles"]}
```

This abstraction, the RL-based policy hardener provides the same protection for all providers, so it is great for any cloud type or setting.

6.3 Policy Anomaly Detection

Introducing reinforcement learning into the agent's decision system improved its efficiency by reducing instances where wrong decisions were taken and improving the effectiveness of security actions. In the beginning, the agent mainly worked by reviewing broad permissions and noticing the "*" rule in AWS or "Owner" role in GCP. Still, because cloud systems are not constant, sometimes fixed rules did not work well.

To take care of this, the RL engine relied on using Q-learning which was trained using past rulings and results. The encoded permission type, scope and how often something is used were in the state and the system gave the same given penalty for excessive privileges and failing an activity. After three weeks, the model started to take over 72% of decisions in QA, making those choices with 93% certainty.

As one can see in the code example, if permissions have not been used for more than 30 days, the system shows them as flagged and automatically cuts their scope, unless they are connected to important systems:

```
for permission in user_policy.permissions:
    if is_unused(permission, days=30) and not is_critical(permission):
```

```
    suggest_removal(permission)
```

Because of this approach, the agent could read behavioral log data (such as CloudTrail or Azure Activity Logs) and propose the best ways to reduce privileges. With the help of machine learning, the process of manually going over cases took up 68% less time, so the security team could focus on other important tasks.

6.4 Cross-Cloud Policy

It was also vital that the autonomous agents by themselves could catch and resolve differences in IdAM policy implementations across different clouds. It was often the case that a service could access AWS resources with admin control but just the ability to read in Azure or GCP. Agents started detecting these inconsistencies and assigned them to their inconsistency managers to deal with them.

Figure 6 Reduction on human effort

When NIST 800-53 and CIS were used to control the simulated audits, the agent-made IAM policies surpassed the success of human-made configurations, passing 96.4% of compliance checks in every cloud versus only 67.8% for the hand-coded configurations. Agents made sure to maintain a live dashboard that showed every success in deployment which helped more teams adopt the tools.

Policy-as-Code (PaC), agents could create GitLab merge requests for any updates, so that important areas still needed to be checked by people. Because of this, it now takes only a few minutes to address a security threat instead of an average of 6 hours before. As a result, the attack surface of the organization went down, mainly because temporary testing environments were not visible and did not have proper governance before.

These outcomes prove that autonomous security agents in DevOps help effectively and scalably ensure that IAM policies are secure in today's multi-cloud world.

Conclusion

It has been proven that autonomous security agents can toughen IAM policy management in diverse DevOps environments involving more than one cloud solution. The combination of constant monitoring, up-to-date enforcement and AI allows the system to greatly lessen the risk of exposing data due to mistaken identities or incorrect permissions.

Numeric outcomes revealed that there were obvious improvements in the time needed to respond, the degree of staff following the rules and how well the setup worked. By using this approach, security and improved speed in

DevOps can both be achieved. Upcoming work will include more agents using federated learning and better CI/CD platform integration to build stronger and recovering cloud security frameworks.

References

[1] Wilde, N., Eddy, B., Patel, K., Cooper, N., Gamboa, V., Mishra, B., & Shah, K. (2016). Security for DEVOPs Deployment Processes: Defenses, risks, research directions. *International Journal of Software Engineering & Applications*, *7*(6), 01–16. https://doi.org/10.5121/ijsea.2016.7601

[2] Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology*, *108*, 65-77. https://doi.org/10.48550/arXiv.1807.04872

[3] Indu, I., Anand, P. R., & Bhaskar, V. (2018). Identity and access management in cloud environment: Mechanisms and challenges. *Engineering Science and Technology an International Journal*, *21*(4), 574–588. https://doi.org/10.1016/j.jestch.2018.05.010

[4] Jambunathan, B., & Kalpana, Y. (2018). Design of devops solution for managing multi cloud distributed environment. *Int. J. Eng. Technol*, *7*, 637-641. https://petacms.io/wp/wp-content/uploads/2022/02/Design_of_devops_solution_for_managing_multi_cloud.pdf

[5] Slawik, M., Blanchet, C., Demchenko, Y., Turkmen, F., Ilyushkin, A., De Laat, C., & Loomis, C. (2017, December). CYCLONE: the multi-cloud middleware stack for application deployment and management. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 347-352). IEEE. 10.1109/CloudCom.2017.56

[6] Sikeridis, D., Papapanagiotou, I., Rimal, B. P., & Devetsikiotis, M. (2017). A Comparative taxonomy and survey of public cloud infrastructure vendors. *arXiv preprint arXiv:1710.01476*. https://doi.org/10.48550/arXiv.1710.01476

[7] Kim, M., Mohindra, A., Muthusamy, V., Ranchal, R., Salapura, V., Slominski, A., & Khalaf, R. (2016). Building scalable, secure, multi-tenant cloud services on IBM Bluemix. *IBM Journal of Research and Development*, *60*(2-3), 8-1. 10.1147/JRD.2016.2516942

[8] Debroy, V., Miller, S., & Brimble, L. (2018, October). Building lean continuous integration and delivery pipelines by applying devops principles: a case study at varidesk. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 851-856). https://doi.org/10.1145/3236024.3275528

[9] Kumar, T. V. (2015). CLOUD-NATIVE MODEL DEPLOYMENT FOR FINANCIAL APPLICATIONS. https://philpapers.org/rec/VARCMD

[10] Ghantous, G. B., & Gill, A. Q. (2018, July). DevOps reference architecture for multi-cloud IoT applications. In *2018 IEEE 20th Conference on Business Informatics (CBI)* (Vol. 1, pp. 158-167). IEEE. 10.1109/CBI.2018.00026

[11] Dijk, F. W. (2017). *Adopting the Cloud: A multi-method approach towards developing a cloud maturity model* (Master's thesis, University of Twente). https://purl.utwente.nl/essays/73075

[12] Wei, H., & Rodriguez, J. S. (2018, August). A policy based application deployment method in hybrid cloud environment. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* (pp. 93-99). IEEE. 10.1109/FiCloud.2018.00021

[13] Ayebo, I. S. (2017). The Role of Serverless in DevOps and CI/CD Pipelines. https://www.researchgate.net/profile/Iyanu-Ayebo/publication/388178322_The_Role_of_Serverless_in_DevOps_and_CICD_Pipelines/links/678d91d098c4e967fa7074b8/The-Role-of-Serverless-in-DevOps-and-CI-CD-Pipelines.pdf

[14] Mohammed, I. A. (2011). A Comprehensive Study Of The A Road Map For Improving Devops Operations In Software Organizations. *International Journal of Current Science (IJCSPUB) www. ijcspub. org, ISSN*, 2250-1770. https://ssrn.com/abstract=4686570