International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Attribute-Based Management of Secure Kubernetes Cloud Bursting

[1] Shaik Uzair ,[2] Mohd Abdul Rahman, [3]Abdul Azeez , [4]Dr.Syed Asadullah Hussaini

[1,2,3]B.E Student, Department of CSE, ISL Engineering College, Hyderabad, INDIA

[4]Assistant Professor, Department of CSE, ISL Engineering College Hyderabad, INDIA.

ABSTRACT:

In modern cloud computing, the need for flexible and scalable orchestration of services, combined with robust security measures, is paramount. In this paper, we propose an innovative approach for managing secure cloud bursting in Kubernetes, combining Attribute-Based Encryption (ABE) with Kuberneteslabeling. Our model addresses the challenges of complexity, cost, and data protection compliance by leveraging both Kubernetes and ABE. We introduce an attribute-based bursting component that uses Kubernetes labels for orchestration, and an encryption component that employs ABE for data protection. This unified management model ensures data confidentiality while enabling efficient cloud bursting. Our approach combines the strengths of label-based orchestration with fine-grained encryption, providing a technologically advanced yet user-friendly solution for secure cloud bursting. We present a proof-of-concept implementation that demonstrates the feasibility and effectiveness of our model. Our approach offers a unified solution that complies with security and privacy laws while meeting the needs of contemporary cloud-based systems.

Keywords:

Role based access control, policy enforcement, Security policies, environmental attribute

INTRODUCTIOION:

In recent years, the proliferation of virtualization and containerization technologies has led to a significant increase in the complexity of distributed systems, as cloud computing systems. As organizations strive to achieve efficient resource management and scalability, Kubernetes has emerged as the most popular solution for orchestrating resources in such complex systems. For example, it is integrated into platforms such as Amazon Elastic Kubernetes Service, Google Kubernetes Engine, Azure Kubernetes Service IBM Cloud Kubernetes Service, and Oracle Container Engine for Kubernetes (OKE). This paper aims at exploring the challenges associated with cloud bursting, which allows private cloud services to use public cloud resources when local resources are exhausted or for any other reasons. Specifically, we address the configuration issues associated with service request load management over a hybrid cloud system including both private and public components. The orchestration of such a heterogeneous system presents a number of challenges, such as optimal management of typically large volume of resources, variable operating conditions, security issues, and compliance with local regulations. In order to address these challenges, resorting to attribute-based management policies is regarded as a valid approach. Attributes are intrinsic characteristics or properties related to the entities, workload, or resources to manage. They can refer to different aspects, such as computational requirements, security levels, data location, and other relevant information. Recent findings indicate that the related management challenges can be effectively addressed through the utilization of an attribute-based approach, which has been found to be preferred over the conventional role-based methods. Rather than depending solely on pre-defined roles, attributes can include a broader array of qualities and attributes associated with users, resources, or data. This level of flexibility, adaptability, and precision in access control renders them more suitable for scenarios characterized by a diverse, dynamic, and intricate range of access requirements. In particular, in this paper, we leverage the attribute-based approach to easily orchestrate load distribution and resource allocation between private and public clouds during the cloud bursting process. Attribute-based policies can be enforced by different technologies. In this paper we make use of Kubernetes, since today it is one of the most appreciate tools for managing distributed systems, especially in the context of cloud computing. It provides flexibility though different built-in components and tools. Among them, the usage of

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP369-373

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

labels and label selectors can be exploited to simplify cloud bursting operations. While Kubernetes best practices recommend that labels be assigned semantic meanings before being used there is currently no standardized method for enforcing this practice. Our goal is to develop a systematic approach in the context of cloud bursting that ensures semantic meaning associated with the generic Kubernetes label concept. Furthermore, it emerged that Kubernetes management does not suitably address all the security aspects related to data confidentiality and access controls, which are central in cloud bursting. Kubernetes incorporates access management, but it requires separate configuration processes that are decoupled from the logic of the orchestrated functions. Moreover, the existing access management mechanisms in Kubernetes have certain limitations in terms of managing complex authorization scenarios and are constrained by their policy scope. Hence, these limitations are challenging for achieving comprehensive and secure resource management in the context of cloud bursting. To overcome these limitations, in this paper we propose an architectural solution to address the security challenges of cloud bursting that integrates the Kubernetes orchestration with attribute-based encryption. When it is necessary to move data to the cloud, it is critical to ensure security and flexible, granular control over file access. This can be efficiently done through ABE. However, user revocation is a significant issue in ABE. In, the authors propose a ciphertext-policy ABE (CP-ABE) scheme with efficient user revocation for cloud storage system. User revocation is handled by introducing the concept of user group, with the rule of updating private keys of the users remaining in the group when any other user leaves it. In addition, since the computation cost of CP-ABE grows linearly with the complexity of the access structure, in order to mitigate it they propose to offload high computation demand to cloud service providers without leaking file content and secret keys. They prove that the proposed scheme can withstand collusion attack performed by the revoked users cooperating with the remaining ones. A similar approach, which requires the update of the unrevoked users' keys, is proposed in It is based on the use of a group manager to accomplish this task. It also applies re-encryption technology to prevent the revoked users from decrypting ciphertexts. However, since the correctness of outsourcing computing results is difficult to guarantee, this approach often requires resorting to the blockchain technology for obtaining such guarantees. Blockchain is used also in [41] to solve the key escrow problem by replacing the traditional key authority with a blockchain. Keys are generated collaboratively by users and the blockchain, thus preventing the latter from obtaining them alone. Alternatively, multi-authority solutions can be used to securely delete data in cloud [48], where data sharing policies can be a challenging issue

METHODOLOGY:

- **User Interface Design**

To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password, Email id, City and Country into the server. Database will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page. It will search the query and display the query.

- **Key Repository**

Key Repository is the second module in our project, where crucial functional requirements of the project will be carried out. KRfirst login with name and password then verify all userrequest for data. Key Repository will share keys to user to access the plain data. When Owner uploads data the keys will be shared Key Repository.

Authority

This is the third module in our project where Authorityplays the main part of the project role. Authority login first then its checks User Registration data, if Authority approve keys for user then he access data or perform an remaining operation.

- **Data Owner**

This is the fourth module in our project where data ownerplays the main part of the project role. Owner register and then login in to the application, the registration details are stored inside database. After Owner Login he will directly navigate owner home page and Upload data. When data owner upload data the data will be encrypted the encrypted keys will be stored inside database and keys will shred with key repository.

- **Data User**

This is the Fifth module in our project where data User plays the main part of the project role. User register and then login in to the application, the registration details are stored inside database. After User Login he will directly navigate User home page and Access data by searching with keyword. When data owner upload data the data will be encrypted the encrypted keys will be stored inside database and keys will shred with key repository.

MAP:

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP369-373

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

THEREAD:

Simply put, a *thread* is a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

CREATING THREADS:

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

TESTING:

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Performance Test

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

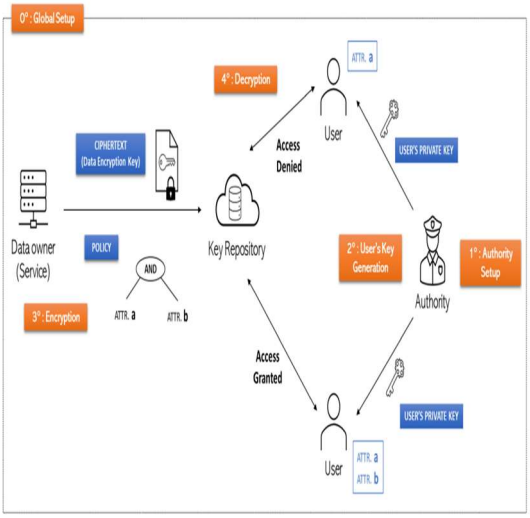
DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP369-373

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

System Architecture



SCOPE OF THE PROJECT

This project focuses on designing, implementing, and evaluating a secure cloud bursting solution within Kubernetes by integrating Attribute-Based Encryption (ABE) and Kubernetes labeling. The primary objective is to develop an attribute-based bursting component that utilizes Kubernetes labels for efficient orchestration of services, and an encryption component that employs ABE for robust data protection.

CONCLUSION:

this paper, we introduce a robust orchestration scheme tailored for secure cloud bursting. This scheme addresses the complexities, cost challenges, and stringent data protection compliance requirements by harnessing the combined capabilities of K8s and ABE. By incorporating ABE, our approach achieves granular encryption and provides cloud resources with suitable confidentiality. At the same time, cloud bursting empowers the extension of computational tasks beyond the scope of a local primary cloud environment. The synergy of these two technologies establishes a cohesive management framework, guaranteeing secure access to bursting services and streamlined deployment of excess workloads to the cloud, all facilitated by Kubernetes.

APPLICATIONS:

FUTURE ENHANCEMENT:

Future research will consider the integration of the proposed solution with artificial intelligence algorithms to proactively infer the need of resorting to bursting operations. In fact, purely relying on a reactive approach the latency of the process can either lead to temporary violation of service level agreements - loose approach - or allocation of excessive resources - conservative approach.

REFERENCE

[1] (Amazon Web Serv., Inc., Seattle, WA, USA). Amazon Elastic Kubernetes Service (Amazon EKS).Accessed: Jun. 8, 2023.

[2] (google, Mountain View, CA, USA). Google Kubernetes Engine.Accessed: Jun. 8, 2023.

[3] (Int. Bus. Mach. Corp., Armonk, NY, USA).IbmKubernetes Service. Accessed: Jun. 8, 2023

[4] "Module aw11 rabe." Accessed: Jan. 5, 2024

[5] (Oracle Comput. Softw.Co., Austin, TX, USA).Oracle Cloud Native Services—Container Engine for Kubernetes.Accessed: June 8, 2023.

[6] "Security best practices for Kubernetes deployment." Mar. 2019. [Online]. Available: https://kubernetes.io/blog/2016/08/security-bestpractices-kubernetes-deployment/

[7] "Anthos." Dec. 2023. [Online]. Available: https://cloud.google.com/ anthos [8] (Palo Alto Networks, Inc., Santa Clara, CA, USA). CloudNative Applications Protection Platform. Dec. 2023.

[9] "Configuration best practices: Using labels." Jul. 2023. [Online].

[10] (CloudBeesSoftw. Co., San Jose, CA, USA).Configuring Cloudbees Build Acceleration for Agent Cloud Bursting. (Dec. 2023).

[11] "Kubernetesautoscaler." github. Dec. 2023.

[12] (Microsoft Corp. Technol. Corp., Redmond, WA, USA). Securing Kubernetes Workloads In Hybrid Settings With Aporeto. (Dec. 2023

[13] (Amazon Web Serv., Inc., Seattle, WA, USA). TLS-Enabled Kubernetes Clusters With ACM Private CA and Amazon EKS, (Dec. 2023).

[14] "Virtual kubelet." Dec. 2023

[15] R. Ahuja and S. K. Mohanty, "A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage," IEEE Trans. Cloud Comput., vol. 8, no. 1, pp. 32–44, Mar. 2020.

[16] S. Ameer, J. Benson, and R. Sandhu, "An attribute-based approach toward a secured smart-home IoT access control and a comparison with a

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP369-373

ISSN 2347–3657
Volume 13, Issue 2s, 2025

role-based approach," Information, vol. 13, no. 2, p. 60, 2022.
[17] D. Balouek-Thomert, E. G. Renart, A. R. Zamani, A. Simonet, and M. Parashar, "Towards a computing continuum: Enabling edge-tocloud integration for data-driven workflows," Int. J. High Perform. Comput.Appl., vol. 33, no. 6, pp. 1159–1174, 2019.
[18] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 1–21, Apr. 2019.
[19] M. Bellare, B. Waters, and S. Yilek, "Identity-based encryption secure against selective opening attack," Cryptol.ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2010/159, 2010.
[20] P. Benedetti, M. Femminella, G. Reali, and K. Steenhaut, "Reinforcement learning applicability for resource-based autoscaling in serverless edge applications," in Proc. IEEE Int. Conf. Pervasive Comput.Commun.Workshops Other Affil. Events (PerCom Workshops), 2022, pp. 674–679.
[21] S. Benitez. "Meet rocket." Accessed: Jan. 5, 2024.[Online].

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP369-373