International Journal of Leading Research Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Cloud Integration for Legacy Application

Binoy Kurikaparambil Revi

Independent Researcher, USA
binoyrevi@live.com

Abstract

In today's world, countless computer applications are currently hosted on on-premises (OnPrem) servers. Many of these applications are well-suited for migration to the cloud, where they can benefit from enhanced security features, upgraded hardware capabilities, and scalable resources that can accommodate fluctuating demand. The transition to cloud environments often leads to increased efficiency, reduced operational costs, and improved performance.

However, a significant number of applications remain anchored to OnPrem systems for various reasons. Data security concerns are paramount, as many organizations have stringent regulations governing the protection and privacy of sensitive information. Additionally, some applications exhibit strong dependencies on OnPrem infrastructure, making migration a complex and risky endeavor. The sheer size of certain applications can also pose challenges. Furthermore, the financial implications of utilizing cloud resources can be prohibitive, especially if the required resources exceed what is available on-site.

In light of these challenges, cloud integration emerges as an option in addition to cloud migration. This approach allows organizations to integrate their existing OnPrem applications with cloud storage and services, enabling them to leverage the benefits of the cloud without the need to move their applications off-site. With cloud integration, companies can enhance their applications' functionality, improve data access, and ensure seamless interoperability, all while maintaining their OnPrem infrastructure.

Keywords: Cloud Integration, Legacy Application, Restful Services

Introduction:

To successfully integrate the application into a cloud environment, it is essential that the application supports the implementation of a secure communication interface. This interface should be capable of facilitating secure communication protocols, such as RESTful services[2], to ensure seamless interaction between on-premises systems and cloud resources. For most of the programming languages and techniques, numerous communication interface protocol development packages are readily available for interface developement. These packages simplify the process of implementing secure and efficient communication channels, making it easier for developers to connect their applications with cloud services.

Problem Description:

Although investing in on-premises services may seem like a one-time capital expenditure, there are numerous factors that raise concerns about the long-term security of data stored on these local servers

and systems. While cloud solutions don't require a large upfront investment, they often involve recurring payment models, which can be daunting, particularly when those costs add up significantly. Despite this, cloud services[3] effectively fulfill their purpose by providing enhanced security and flexibility that on-premises solutions may lack. Major problems On-Prem services are facing are listed below:

1. Huge Capital Investment: Setting up the necessary infrastructure often involves more than just hardware devices and accessories. It requires substantial resources such as server rooms, building space, a power supply, air conditioning, and physical access security systems, which means a significant initial investment is needed. Additionally, implementing a backup strategy can further increase this capital expenditure. However, if cloud migration or integration is done, these things can be done in just a matter of minutes, and no infrastructure needs to be owned or maintained.
2. Security Updates: Delays in applying security updates and patches to on-premises servers and systems may pose potential risks. In contrast, cloud infrastructure is regularly updated with the latest security patches and updates. Furthermore, most cloud services provide double encryption[1] for the data stored in the cloud.
3. Upgrades and Downgrades are complex: On-premises systems are difficult to upgrade due to downtime, labor, and hardware costs. However, cloud infrastructure is easy to scale according to the needs.
4. Backup plans are costly: On-premises servers are difficult to manage when taking backups, especially in a completely different geographic area if a risk mitigation plan dictates that. However, cloud infrastructure provides high data availability and makes implementing risk mitigation easier.

Legacy applications often present significant challenges that stem from a variety of constraints, including strict domain regulations, security standards, and specific data handling protocols. These applications were not originally designed with cloud integration in mind, which creates additional hurdles for modernizing them. Consequently, even if there is an intention to migrate these legacy systems to the cloud in the future, such transitions may not be feasible due to these inherent limitations.

Cloud Integration using RESTFul Services:

Many legacy applications have been built using programming languages such as C/C++ and Java. Within this category, many intricately designed backend applications that handle complex data processing[4] are primarily developed in C/C++. Transitioning these applications to modern programming languages often poses significant challenges, as the rewrite process can extend over several years and typically results in numerous cycles of bug fixing. Furthermore, the performance of C++ applications is still considered best in many backend use cases. So integrating these legacy systems with cloud solutions presents a more manageable and efficient approach to data management.

One of the most reliable and secure methods for achieving this integration is using RESTful services[2]. QT, a robust C++ framework, offers a streamlined way to develop communication modules that can interface efficiently with these legacy applications and the cloud. By leveraging QT, developers can create solutions that bridge the gap between outdated software and modern cloud infrastructure, ensuring continued functionality and secure communication to manage data on the cloud.

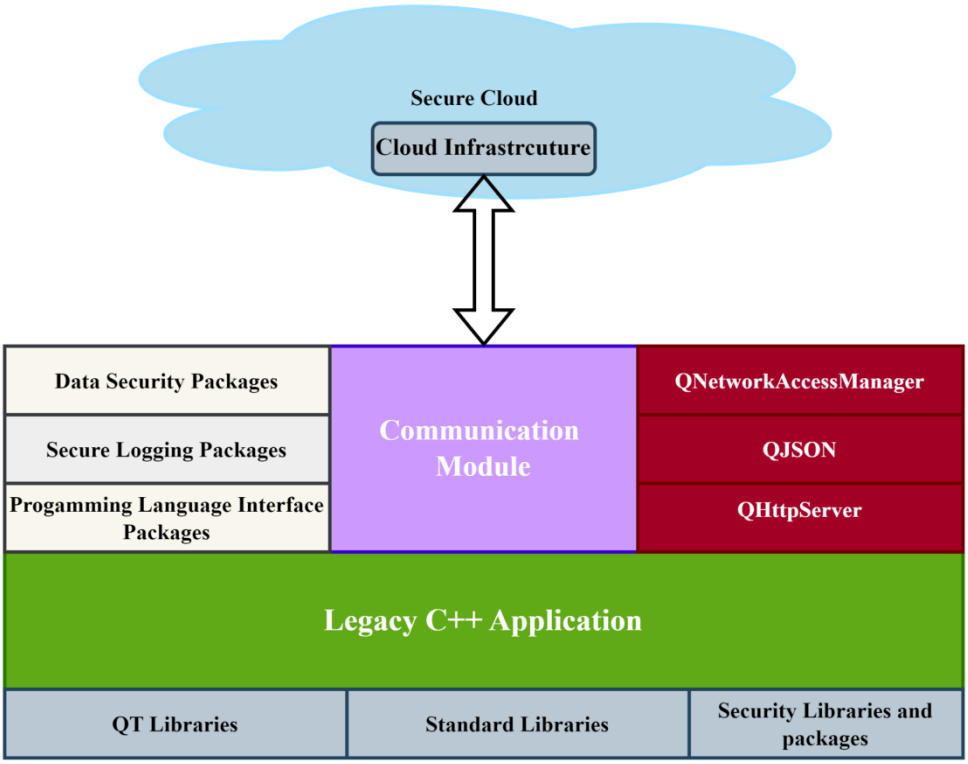

Figure 1: Architecture for Backend Application Integration to the Cloud.

Figure 1 presents a comprehensive high-level architecture integrating the C++ application with cloud services. At the heart of this architecture lies the communication module, a robust component developed in C++ utilizing the Qt framework and its associated libraries. This module is crucial in facilitating efficient data exchange between the application and the cloud. Notably, the architecture employs a JSON data model, a widely used format for data interchange in the cloud and web application domain, for various reasons. The following points delve into the specifics of the different components that make up this architecture, highlighting their functions and contributions to the overall system.

1. QT Libraries - These are the core libraries for the application development using the QT framework. It's important to note that QT provides a wide range of libraries, and development should be narrowed down to QT Core and necessary QT libraries.
2. Standard and Security Libraries - Using the latest stable and secure versions of these libraries is crucial, as the legacy application cannot function without them. The security libraries include code security libraries like smart pointers and data security libraries like SSL.
3. QJSON - Any standard JSON package should work, but I prefer QJSON because it is available in the Qt framework. The BOOST library is another good option. The JSON data format is widely used across cloud and web technologies and is generally considered more secure than the XML format, which was commonly used in the industry in the past. This makes JSON an ideal choice for designing the data model.
4. QT Network Libraries - The QHttpServer and QNetworkAccessManager are the spotlight packages that help the communication module communicate with the secure cloud using Restful Services[2] and the JSON data model. This handles the HTTP request and response.

5. Data Security Packages - The data security packages contain the essential libraries necessary for data encryption and decryption[1], both for data at rest and during transfer.
6. Secure Logging Package - This situation is a bit tricky because the legacy application is already using a specific package for its logging. Therefore, continuing to use that package or upgrading to a more secure version should be the top priority. However, we can also consider upgrading to a completely new, more secure, and sophisticated logging package, as the logging communication module is critical.
7. Programming Languages Interfaces - Legacy applications can be developed in various programming languages. Pybind11 is a package that facilitates the creation of an interface between C++ and Python code can be considered as an example.
8. Communication Module - This innovative module has been designed to connect legacy applications with cloud environments through the use of Restful Services[2]. A key element of this development process is the emphasis on utilizing pre-existing packages that have been previously mentioned rather than attempting to create new libraries from scratch. This approach is crucial because these established packages are not only mature and thoroughly vetted but also possess robust security features. Attempting to write custom code for these functionalities can lead to significant challenges and potential risks.
9. Cloud Infrastructure - Cloud infrastructure is a set of cloud services that are tied together to perform a set of tasks. In this case, the cloud infrastructure is set up to handle the HTTP communication via Resful services, data retrieval, data storage, secure virtual network, Database and storage services, etc. Care must be taken to design this each service has a cost associated with it and the good thing is that most services are scalable.

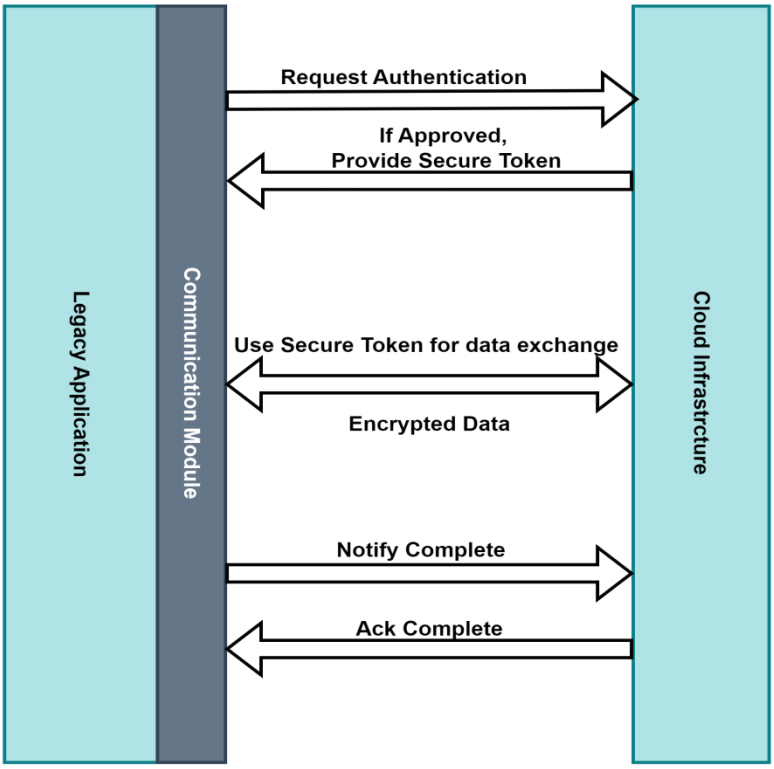

Figure 2: Communication protocol for secure data exchange

Figure 2 illustrates a typical protocol used for data exchange between the communication module and the cloud infrastructure. The process begins with the communication module sending the necessary encrypted data[1] for authentication. The cloud service[3] validates this request and responds with a temporary token, which expires after a predefined period.

After receiving the token, the data exchange process commences. The communication module includes the token in every HTTP POST request sent to the cloud as part of the RESTful services[2]. The cloud service then responds with either a status update or the requested data. Finally, the connection is automatically terminated after the token expires or when the communication module sends a "Notify Complete" request.

Conclusion:

It's always a challenge for businesses and organizations to reuse valuable legacy code while also benefiting from modern cloud computing[4] and storage. Many struggle to transition confidently to modern programming languages and techniques due to the substantial domain knowledge and expertise embedded in their legacy systems. These legacy codebases often encompass millions of lines of code that implement numerous options, algorithms, and logic. Translating all of this code into a new language cannot be accomplished overnight. However, the demands for improved memory and performance drive these organizations to adopt cloud solutions, allowing them to integrate legacy applications and leverage significant advantages in terms of memory and computing power. RESTful services[2] play a key role in this implementation. The RESTful services[2] allow the organization to modernize the User Interface (UI) screens using technologies like React JS. This results in sleek, reliable, fast, and visually appealing screens without altering the backend legacy application.

References:

1. Q. Zhang, "An Overview and Analysis of Hybrid Encryption: The Combination of Symmetric Encryption and Asymmetric Encryption," 2021 2nd International Conference on Computing and Data Science (CDS), Stanford, CA, USA, 2021, pp. 616-622, doi: 10.1109/CDS52072.2021.00111.
2. Neumann, N. Laranjeiro and J. Bernardino, "An Analysis of Public REST Web Service APIs," in IEEE Transactions on Services Computing, vol. 14, no. 4, pp. 957-970, 1 July-Aug. 2021, doi: 10.1109/TSC.2018.2847344.
3. Alessio Botta, Walter de Donato, Valerio Persico, Antonio Pescapé, Integration of Cloud computing and Internet of Things: A survey, Future Generation Computer Systems, Volume 56, 2016, Pages 684-700,
4. ISSN 0167-739X, https://doi.org/10.1016/j.future.2015.09.021.
5. Manuel Díaz, Cristian Martín, Bartolomé Rubio, State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing, Journal of Network and Computer Applications, Volume 67, 2016, Pages 99-117, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2016.01.010.
6. Christos Stergiou, Kostas E. Psannis, Byung-Gyu Kim, Brij Gupta, Secure integration of IoT and Cloud Computing, Future Generation Computer Systems, Volume 78, art 3, 2018, Pages 964-975,ISSN 0167-739X, https://doi.org/10.1016/j.future.2016.11.031.