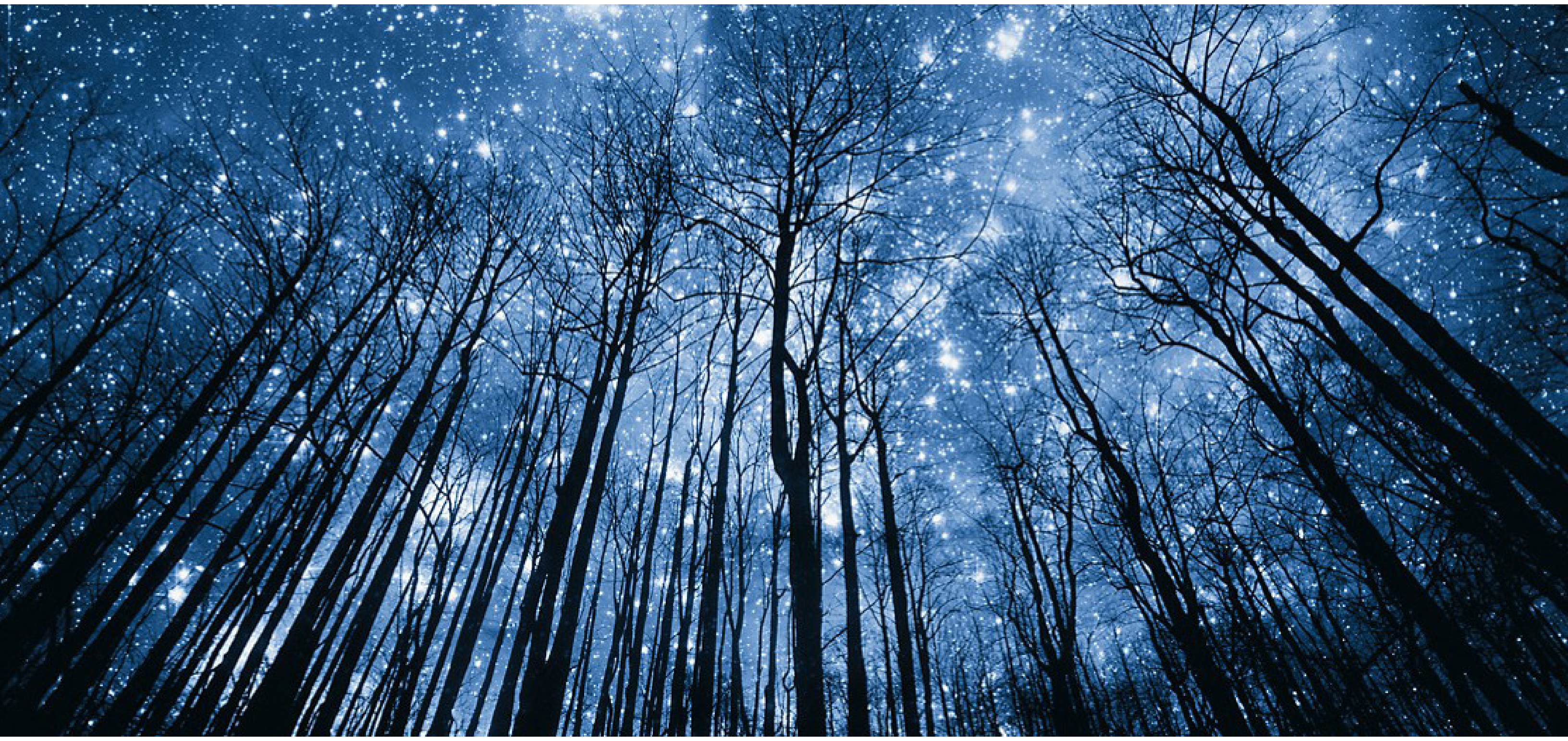
IMPORTANCE OF CLOUD MIDDLEWARE TECHNOLOGIES



Prateek Bhat
Inside Product Specialist
Dell Technologies

Knowledge Sharing Article
© 2023 Dell Inc. or its subsidiaries.

The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across Dell's multiple technologies and products with both skill and outcome-based certifications.

Proven Professional exams cover concepts and principles which enable professionals working in or looking to begin a career in IT. With training and certifications aligned to the rapidly changing IT landscape, learners can take full advantage of the essential skills and knowledge required to drive better business performance and foster more productive teams.

Proven Professional certifications include skills and solutions such as:

- Data Protection
- Converged and Hyperconverged Infrastructure
- Cloud and Elastic Cloud
- Networking
- Security
- Servers
- Storage
- …and so much more
-

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

TABLE OF CONTENTS

ABSTRACT

We all know that organizations strategical approach for application development, storage and other services have prodigiously transposed over the decade. Cloud is one of the transcendent technologies that enables and empowers these strategies. Cloud-native is a methodical technique of developing and running applications that utilize cloud computing approach without locking-in to a specific cloud platform. Cloud-native technologies enable firms, companies, and organizations with cloud computing delivery models to develop and run applications on dynamic, scalable, secure domains such as public, private and hybrid clouds also enabling increased scale, resiliency, and speed. It is expected that the value of the public cloud market will hit $623.3 billion by 2023 and more than 70% of global organizations expect to be hosting greater than two containerized applications in production, which has inflated from 20% in 2019.

It is important that all these technologies are interconnected. There are lot of middleware technologies that are encapsulated under these systems to facilitate smooth data transition and transport to all the different layers of the stack. Middleware is the software that connects software components or enterprise applications in a distributed system. Similarly, a cloud middleware is software that acts as a liaison between applications and networks. Some examples where middleware technologies are used prominently include Enterprise Application Integration software, telecommunications software, transaction monitors, and messaging-and-queueing software, cloud computing, and interconnected IOT or communication devices.

This article depicts the importance of cloud middleware technologies and how they are implemented in cloud systems to facilitate connection, networking, and data transfer. The article also emphasizes on use cases and application of such technology in the real world. This will also cover the advantages and challenges that can be observed in the systems before implementing the cloud middleware and after implementing it.

Distributed Transaction Introduction

Before we encounter the middleware technologies and its related topics, let us brush through briefly as to what is a transaction as it would be a fundamental phenomenon of a system.
A distributed transaction can be stated as a set of operations that are implemented or performed on data entities, across various systems or a single monolith system. Especially, the distributed transactions are common phenomena in a typical database environment. Generally, these transactions are coordinated across separate systems or nodes connected by a network, however, they may also span across multiple databases on a single server. Normally, a transaction symbolizes a unit of work performed which might include multiple operations in various situations if needed, within a system either distributed or monolith and treated in a coherent and reliable way independent of other operations.

Generally, these transactions have a specific property that they abide by. These properties are called “ACID Properties.” ACID is an abbreviated version of Atomicity, Consistency, Isolation, and Durability. Briefly stated:

- **Atomicity:** Atomicity ensures that a transaction's constituent instructions are handled as a single entity and will either succeed or fail together. This is crucial so that, in the event of an unfavorable occurrence, such as a breakdown or power loss, we can be certain of the database's condition. If any portion of the transaction failed, it would have either been successfully completed or rolled back. The source is debited of money in the case above, and if an anomaly arises, the modifications are thrown out and the transaction fails. To make sure that the IPA systems are being utilized efficiently and morally, it is crucial to have good governance and monitoring in place.
- **Consistency:** Consistency ensures that modifications performed inside a transaction comply with database restrictions. This covers all guidelines, limitations, and triggers. The entire transaction fails if the data enters an unlawful condition. Let us imagine there is a requirement that the balance must be a positive integer, returning to the previous example of a money transfer. The balance will not satisfy the limitation if we try to take out more money than is allowed. As a result, the ACID transaction's consistency will be broken, and the transaction will fail.
- **Isolation:** All transactions are guaranteed to run in an isolated environment thanks to isolation. This makes it possible to conduct many transactions simultaneously because they do not conflict. Let us imagine, for illustration, that the amount in our account is $200. At the same moment, two withdrawal operations totaling $100 begin. Since the transactions are carried out independently, after they are both finished, we will have a balance of $0 rather than $100.
- **Durability:** Durability ensures that changes are persisted after the transaction has finished and the database has been updated. As a result, even in the event of system failures like crashes or power outages, the data within the system will continue to exist. Developers may make intricate, coordinated modifications, and rest easy at night knowing that their data is reliable and securely kept thanks to the ACID properties of transactions.

Together, these ACID characteristics make sure that, even in the case of unanticipated mistakes, a series of database operations (grouped into a transaction) leaves the database in a valid state.
The core of a computer system is this class of distributed transactions. They finally assess the system's effectiveness. Organizations aim to accomplish the efficient implementation of these transactions.

Introduction to Middleware

With emerging technologies in the industry, connected devices, data transfer and various other activities are common in different sectors. We can clearly see the applications used, interact with them to collaborate, create, manipulate data, in-order to achieve the desired result. However, we might not be able to see the technologies that are behind the screen, in the sense, the enabling technologies that might be involved behind these applications to empower them do what is desired. These types of technologies are called “Middleware.”

It is stated that – “The software that links corporate applications or software components in a distributed system is known as middleware.” This is accurate, however it is possible that certain pieces of hardware and software connect computers and other gadgets to other applications. It may also be referred to as the client/server system or connecting point. Another approach to describe middleware is as software that serves as a bridge between networks and applications. Middleware is, to put it simply, a software platform that stands in between one application or device and another. Although it is not directly utilised by end users, it enables communication between any two clients, servers, databases, or even applications.

The phrase "middleware," which has been used in the context of software engineering since the late 1960s, can refer to a variety of contemporary software components. Application runtimes, business application integration, and numerous cloud services are examples of middleware. Middleware frequently handles data management, application services, communications, authentication, and application programming interface (API) administration.

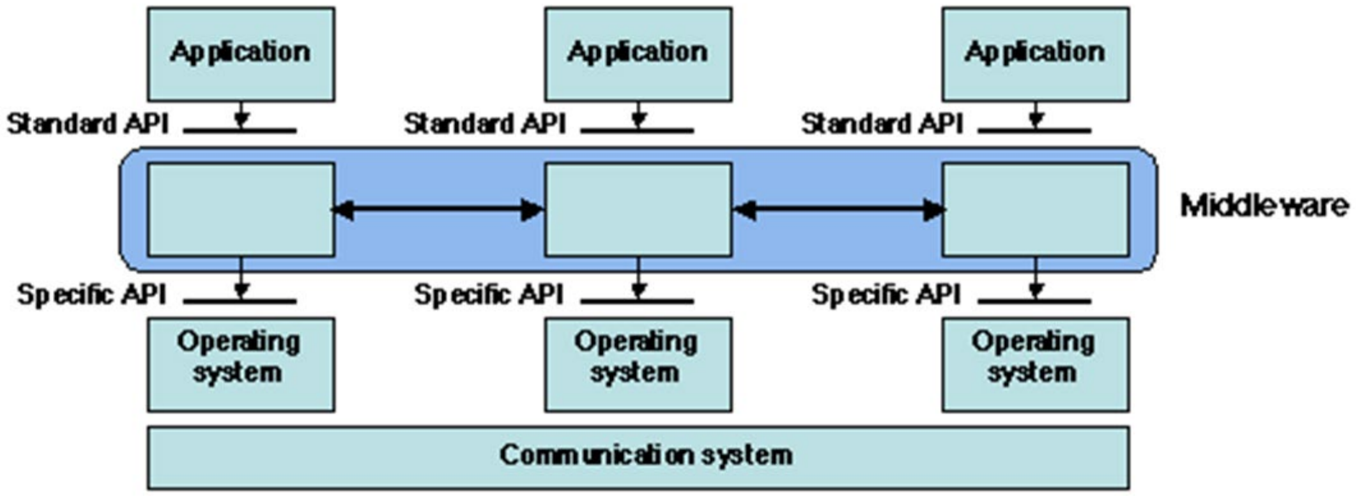


Fig 1: Typical Middleware schematic representation
Source: BITS Pilani course Material

Software called middleware is used by many apps to connect with one another. So that you may develop more quickly, it offers capabilities to integrate apps intelligently and effectively. Middleware serves as a link between various tools, technologies, and databases to enable smooth system integration. Then, the one system offers its consumers a united service. For instance, users of a Windows frontend programme may transmit and receive data from a Linux backend server without realising the difference.

Software and cloud services known as middleware assist developers and operators create and deploy applications more quickly by giving common services and capabilities to apps. The connecting tissue between applications, data, and users is middleware. The technology basis for contemporary cloud-native architectures is middleware today. Middleware can help businesses with multi-cloud and containerized systems create and execute applications at scale economically.

Importance of Middleware

Before it became widely used in the 1980s, middleware served as a link between new applications and legacy systems. Programmers first used it to integrate new programmes with older systems without having to rewrite the old code. In distributed systems, middleware has grown to be a crucial instrument for data management and communication. The technological infrastructure of today is no longer a monolithic standalone system with all components developed and deployed for a single, homogeneous system.

Technology is always changing. Affordable and adaptable cloud computing options have made it possible for companies to advance with this cutting-edge technology. Companies using cloud services have a major competitive advantage thanks to the fast rate of innovation in this industry. IDC anticipates that cloud spending will increase to go over $1 Trillion by 2024as a result of the recent significant growth in cloud adoption rates. The customers of today have advanced alongside technology. They can use any gadget, from a wristwatch to an advanced computer with high setup, to access a company's services. By 2029, more than fifteen billion IoT devices will link to business infrastructure, according to Gartner.

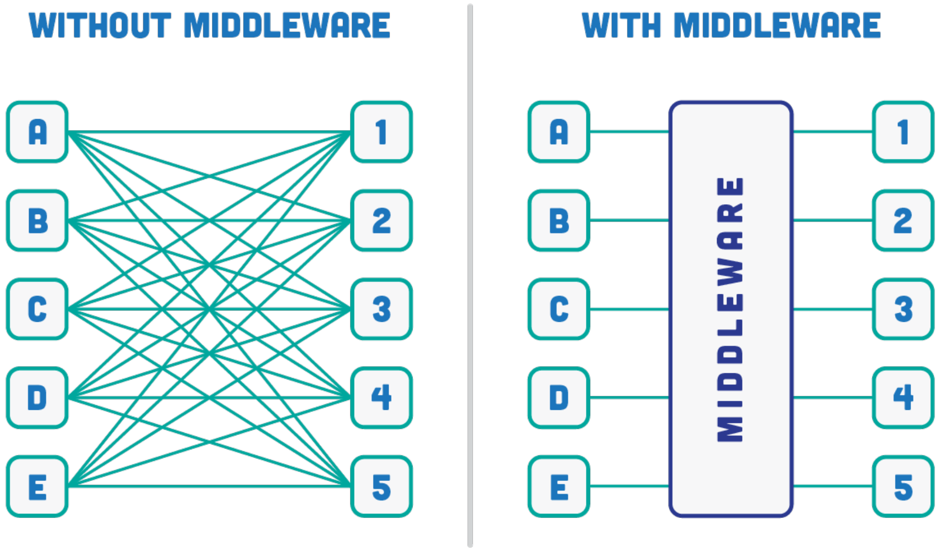


Fig 2: Importance of Middleware
Source: https://blog.briteskies.com/blog/what-is-middleware-and-why-should-you-care

Middleware is a tool used by developers to streamline the design process and facilitate application development. This frees them up to concentrate on features and business logic rather than connection between various software components. For each piece of software that connects to the application, developers would have to create a data exchange module if middleware did not exist. This is difficult since contemporary programmes are made up of several interconnected microservices or tiny software components.
With access to such quickly evolving and diverse technologies, most firms have adopted a "best of breed" strategy, which entails choosing the best product or service to meet each specific business need before figuring out how to make them all function together.

Middleware enters the scene in this situation. Middleware is sometimes referred to as the "glue" that connects dissimilar systems, including legacy infrastructure and edge computing. Each system uses a different programming language as its foundation. What one system comprehends could be unintelligible to another, according to this. Like a translator, middleware makes it possible for individuals from different corners of the world to communicate with one another. It offers a platform for communication that facilitates easy integration and interoperability.

Architecture of Middleware

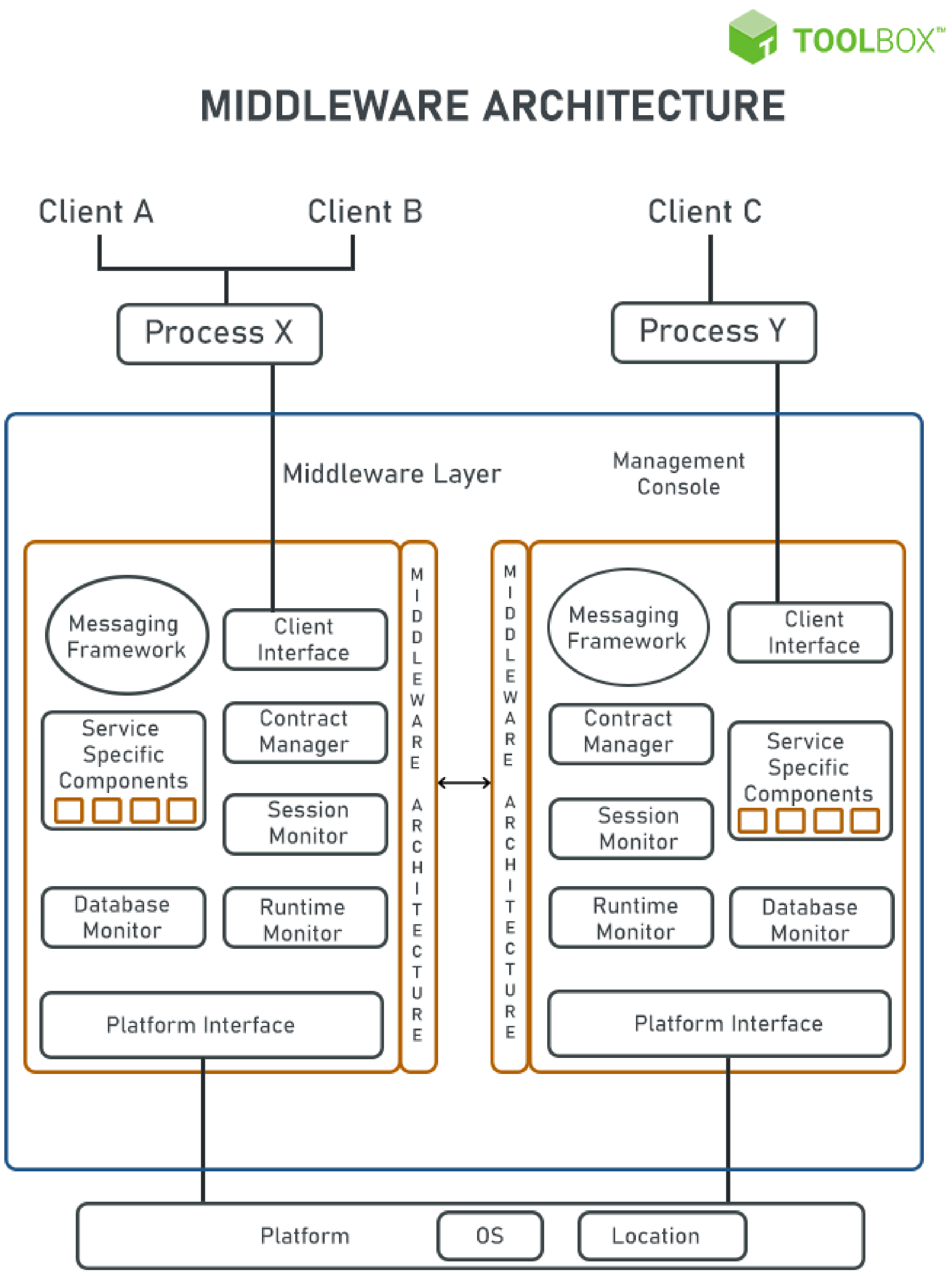


Fig 3: Architecture of Middleware
Source: https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/

The architecture of middleware software is made up of several interconnected parts that work together to form a data pipeline. Through the middleware, the data is sent from one connected application to another. The data is processed by the middleware for compatibility. These are typical elements of middleware:

- Management console
- Client interface
- Middleware internal interface
- Platform interface
- Contract manager
- Session manager
- Database manager
- Runtime monitor

Most middleware systems are software-only, with hardware components appearing only very infrequently. Middleware's purpose is to act as a virtual barrier between different applications and the platform layer. Most of the middleware is either platform-as-a-service (PaaS)-based or has a service-oriented architecture (SOA) design. The goal of the SOA architectural design is to create loosely linked software applications that communicate with one another and function as a unit. Organizations attempting to decouple all their business divisions have embraced it, relying on integration and reusability for everyday operations. Organizations may leverage their current application and system investments thanks to SOA.

Service-oriented design is the clear winner in every middleware need. This necessitates the use of independent, reusable components with unique topologies. Each of these elements must be able to communicate with other elements of the system and one another. Each sort of middleware requires a unique component in addition to a few fundamental ones. For instance, a database management component is required for a middleware for databases.

Now let us briefly have a peak at the key entities of each generic middleware component that is deployed across the system:

1. **Middleware management console**

A single pane of glass called; middleware management console provides a panoramic view of the middleware architecture. With enterprise application middleware, which was created and installed explicitly with an organization's business objectives in mind, this is extremely crucial. An overview of events and activities, transactions, configuration management, and contract rules are provided by this console.

2. **Common messaging framework**

To interface with services, applications, and platforms, middleware needs messaging services. Most of these frameworks rely on already-in-use protocols like JavaScript object notation, REST, and simple object access protocol (SOAP) (JSON). Care must be taken to thoroughly construct this framework, considering both current and future features. Application programming interfaces or web services are used for the actual communication (APIs). Information is received, sent, and transferred across the various layers using standards like JSON, which are used by APIs and web services.

3. **Client interface (or application interface)**

The middleware offers this interface to the client or application services. It enables applications to launch a transaction involving the database, platform, or other backend services using specified structures. One of the major advantages of adopting any middleware solution is the simplicity with which application development and deployment can be done thanks to this component.

4. **Middleware internal interface**

To maintain the overall middleware structure, middleware instances communicate with one another over this interface. Because several middleware instances must cooperate internally to look like a single continuous layer, this component is necessary. The middleware often defines a unique protocol for this purpose that does not conflict with any other communication protocols.

5. **Platform interface**

Regardless matter where it is, middleware needs to function on several platforms. This interface has a direct line of communication with the backend servers. Such type of interface needs to be upgraded to handle new platforms whenever they are released, such those offered by cloud providers. The middleware's other parts are all unaltered.

6. **Contract manager**

Each unit application, data control application, and server impose its own set of rules, which are enforced by a contract manager. To maintain openness and reduce breaches, these contracts must be adhered to across all channels of communication and event planning. The contract manager ensures that the business logic is sound and unbroken. All contract violations are reported to the application, which prevents the system from being disrupted.

7. **Session manager**

The session manager makes sure that all requests for transactions and communications are legitimate and have not expired. In the beginning, it creates the connections necessary for requests to be sent and received. Additionally, the session history is managed for auditing reasons. A session manager is one technique to guarantee the security of the middleware.

8. **Database manager**

The database manager manages database connections and changes data insertion, modification, and deletion depending on the DB service being utilised. Security is a key factor for this component because this much depends on the type of database used (file-based or table-based), where it lives (internal data centre or cloud), and how sensitive the data is.

9. **Runtime monitor**

All contract validations, the runtime monitor monitors session history, requests, and answers. IT administration and security teams utilise it to identify and notify any unusual activities. Additionally, it frequently serves as the foundation for reporting engines that generate audit reports for compliance needs.

Besides these, each type of middleware may require unique services to fulfil its requirements. A device middleware concentrates on the device it caters to. Each of these components is designed and built with a varying focus based on the type. Message-oriented middleware needs a robust common messaging framework, while the database manager is not required to function.

Types of middleware

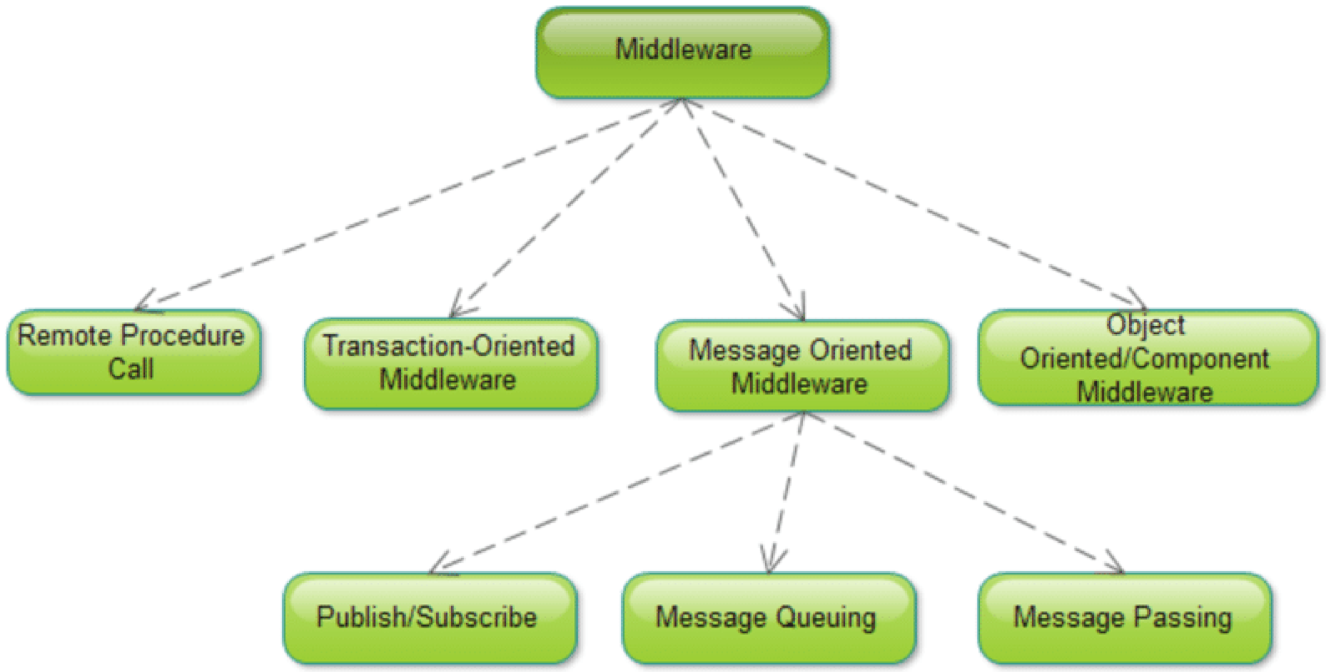

Middleware
Remote Procedure Call
Transaction-Oriented Middleware
Message Oriented Middleware
Object Oriented/Component Middleware
Publish/Subscribe
Message Queuing
Message Passing

Fig 4: Types of Middleware
Source: https://www.researchgate.net/figure/Categories-of-Middleware_fig2_265350963

The word 'middleware' is a broad term covering different types of middleware based on the various functionalities they provide. Some common types of middleware include:

- **Database middleware**: The most popular middleware for enabling access to and interaction with various database gateways is database middleware.
- **Message-oriented middleware:** It enables the exchange of messages between software programmes running on various operating systems and networking protocols.
- **Portal middleware:** Businesses employ portal middleware to simplify communication between front-end and back-end systems.
- **Enterprise application integration**: Enterprise application integration is a virtual layer that links applications, data, processes, and services regardless of their location (the cloud or on-premises) and underlying technology.
- **Transactional middleware:** This kind of middleware makes sure that transactions between various components go through each phase and are completed.
- **Content middleware:** Like publish and subscribe models, this is used to abstract certain content.
- **Remote procedure call (RPC) middleware:** This enables two applications that are not on the same network to communicate with one another and activate each other's functions.
- **Device middleware:** This is a kind that has a particular set of skills for integrating with or creating apps for devices. Typically, it is utilised to create mobile applications.
- **Object request broker (ORB) middleware:** manages requests from one application to another without requiring the apps to care about where each is located.
- **Platform middleware:** Business logic may be placed on any platform (OS or hardware), including web servers, application servers, hosting environments, or containers, thanks to this type of middleware.
- **Application server middleware:** This framework enables the development, distribution, and upkeep of business applications inside a system.

- **Web middleware:** Like the e-commerce example from before, this enables businesses to interact more quickly with unique backend systems.
- **Robotics middleware:** Regardless of the manufacturer or location, this makes it easier to integrate robotic hardware, firmware, and software.
- **Cloud middleware:** Cloud middleware provides a remote platform for developing, managing, and interacting with the hosted applications and data. It resides between the operating systems powering the cloud and the cloud users.

These types of middleware are currently used by most enterprises for daily operations. These are frequently needed in conjunction in large organisations to be financially, operationally, and technologically astute.

What is middleware in cloud computing?

Building and delivering cloud-native apps across various infrastructures is a component of cloud computing. To access cloud resources without being overwhelmed by the difficulty of managing the infrastructures, developers employ middleware. When using a scalable cloud-based hosting platform like Amazon Elastic Compute Cloud, developers execute cloud - based applications in containers (Amazon EC2).

Cloud middleware often sits in between operating system and an application, offering the user a range of features. It supports concurrency, transactions, threading, and messaging; it aids in the development of business applications; it offers a service component architecture framework for the development of service-oriented architecture (SOA) applications. Some examples of cloud-based middleware include databases, web servers, and application servers. Middleware programmes often offer communication services and act as a messenger to enable message sending and receiving between various applications. Through cloud middleware, many apps located at various physical locations can be "connected" together to carry out a task.

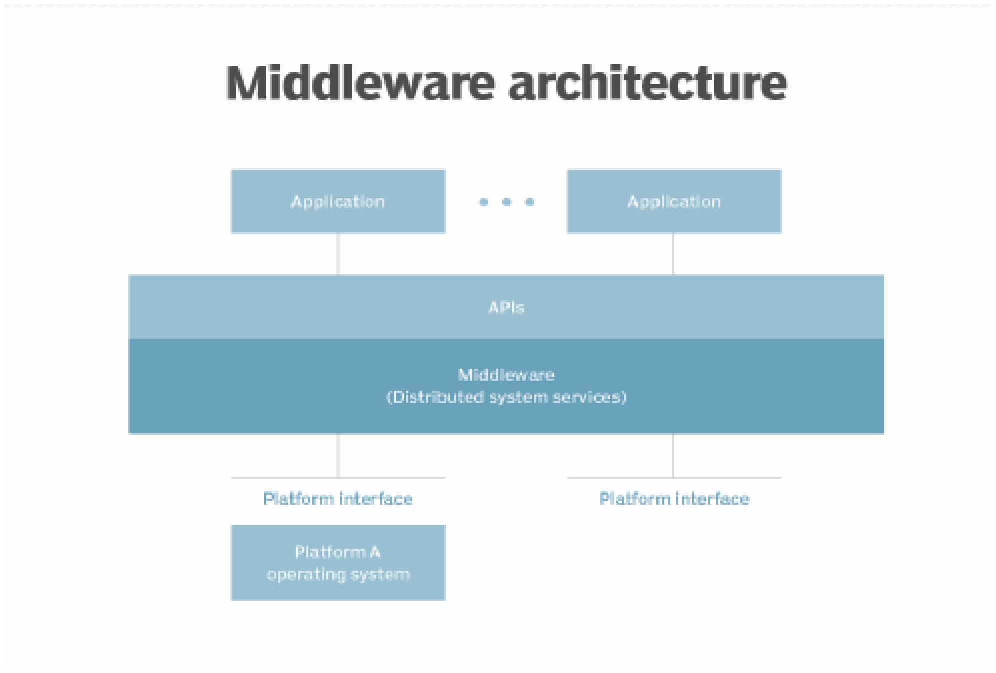


Fig 5: Middleware in cloud
Source: https://www.techtarget.com/searchapparchitecture/definition/middleware

Software called middleware sits between the operating system and the programmes it runs. Middleware serves as a hidden translation layer that facilitates data management and communication for remote applications. Plumbing is another name for it since it joins two programmes together to provide a "pipe" for data and databases. Users can submit forms on a web browser or ask the web server to provide dynamic web pages depending on a user's profile by employing middleware.

Examples of common middleware include transaction-processing monitors, message-oriented middleware, web middleware, database middleware, and application server middleware. Typically, each programme offers messaging services so that multiple programmes may connect with one another utilising messaging frameworks such web services, REST, SOAP, and JavaScript object notation (JSON). The kind of middleware a firm uses will depend on the services being used and the kind of information that must be exchanged, even though all middleware provides communication duties. This can comprise application servers, web servers, message queues, transaction management, security authentication, and directories. In addition to being used for back-and-forth data transfers, middleware may also be used for distributed processing where decisions are made in the present.

Why is middleware important to cloud computing?

Software engineers and systems architects have had to concentrate on the initial design and structure of their technology platforms as enterprises shift more toward cloud-native development. This calls for the selection and configuration of middleware-assisted frameworks and capabilities for the creation, deployment, and operation of applications. A business may reap greater cloud advantages if these tools are in place. Applications may be deployed across a variety of infrastructures, including public clouds and on-premises systems, and continue to function as intended.

Organizations use middleware, most of which is now provided as cloud services, streamlining deployment and maintenance, to manage complexity and maintain rapid and economical application development. On a massively distributed platform, middleware can enable application environments that operate smoothly and reliably.

So, what function does middleware serve in the creation of apps? Modern corporate applications are built to operate in multiple clouds, on-premises, and at scale. Developers want an application environment with unified underlying capabilities to create them. The secret to putting up such an ecosystem is middleware.

We can think of these capabilities in four layers, plus tooling:

- **The container layer**
  This layer of middleware controls how applications are delivered in a consistent way throughout their lifecycles. With CI/CD, container management, and service mesh capabilities, it offers DevOps capability.

- **The runtimes layer**
  The execution contexts for custom code are included in this layer. For highly dispersed cloud systems, such as microservices, in-memory caching for quick data access, and messaging for speedy data transfer, middleware can offer lightweight runtimes and frameworks.

- **The integration layer**
  Through messaging, integration, and APIs, integration middleware connects both bespoke and pre-purchased programmes as well as Software-as-a-Service (SaaS) assets to create functional systems. Data/event streaming, API administration, and in-memory database and data caching services are also available.

- **The process automation and decision management layer**
  Development middleware's fourth and final layer offers crucial intelligence, automation, and decision-making capabilities.
- **Tooling**
  Application development tooling is another layer of middleware in addition to these four. This enables effective code sharing and collaborative development while enabling teams to construct apps using pre-established templates and containers. On-premises and cloud application development and delivery experiences are supported by tooling.

Middleware Benefits

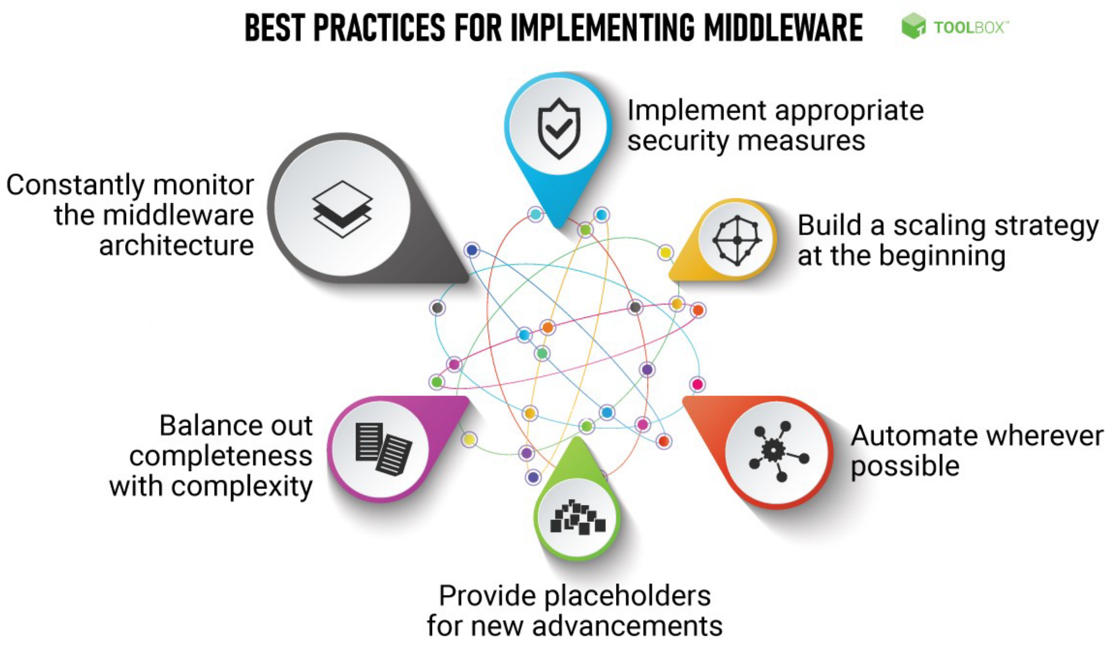

BEST PRACTICES FOR IMPLEMENTING MIDDLEWARE
TOOLBOX
Implement appropriate security measures
Build a scaling strategy at the beginning
Automate wherever possible
Provide placeholders for new advancements
Balance out completeness with complexity
Constantly monitor the middleware architecture

Fig 6: Benefits of Middleware
Source: https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/

Large businesses often boast complicated operations and offer a variety of services to customers in many industries. An organisation may have isolated operations and infrastructure that deviates from its overall strategy if it has purchased multiple smaller businesses. Innovation requires adaptability, even for tiny businesses striving to grow. Each of these cases involves the use of middleware.

The various benefits of middleware for enterprises are:

1. Ensures longevity of legacy investments
2. Allows hassle-free scaling
3. Cuts costs
4. Fosters innovation
5. Manages complexity
6. Automates business processes

Conclusion

So, we can state that Middleware is important to implement different API technology applications, Runtimes, Remote Procedure calls and other cloud native applications as well. A software layer called middleware technology lies in between several programmes and allows for data sharing and communication. It serves as a link between several systems, enabling them to operate together without difficulty even when they were not intended to be compatible.

We have noted that there are several types of middleware, such as: Message-oriented middleware (MOM) and remote procedure call (RPC) middleware are two examples of communication middleware that allow various systems to connect with one another. Transaction middleware that handles and organises transactions across several systems, assuring that they are atomic, consistent, isolated, and durable (ACID). A distributed environment's objects can communicate with one another thanks to an object request broker (ORB). Event-driven middleware makes it possible for systems with event-driven architectures to communicate with one another.

We also visualized the several advantages that middleware technology may offer including like enabling various systems to cooperate in a seamless manner, Bringing some operations, such security and data management, under one umbrella, spreading processes and controlling load to increase performance and scalability, enabling more effective resource management and lowering the requirement for specially constructed system connections, Enterprise systems, distributed systems, Internet of Things, mobile and online apps, cloud computing, and many more systems make extensive use of it.

Overall, we can conclude that, Middleware technology is one such significant technology because it serves as a link between several software programmes, enabling data sharing and communication. This makes it possible for systems that were not designed to communicate with one another to operate together without issue. Additionally, middleware enables the centralization of some operations, such as data management and security, which makes it simpler to maintain and upgrade these features across several systems. Additionally, it can assist in enhancing a system's speed and scalability by controlling load and dispersing processes.

Bibliography

- Cloud Middleware – by Techopedia – Aug 2022 - https://www.techopedia.com/definition/30630/cloud-middleware-software
- Cloud Middleware – ATOS Apperenda - https://apprenda.com/library/glossary/definition-cloud-middleware/
- What Is Middleware? – AWS Middleware Blog - https://aws.amazon.com/what-is/middleware/
- What Is Middleware? Definition, Architecture, and Best Practices – Ramya Mohanakrishnan – Feb 8, 2022 - https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/
- What is middleware? – Azure Blog - https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-middleware/
- What is middleware? – Red Hat – Dec 16, 2022 - https://www.redhat.com/en/topics/middleware/what-is-middleware
- Figures - Luis Lino Ferreira - https://www.researchgate.net/figure/Categories-of-Middleware_fig2_265350963
- https://networkinterview.com/what-is-a-middleware/

Dell Technologies believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

Disclaimer: The views, processes or methodologies published in this article are those of the authors. They do not necessarily reflect Dell Technologies' views, processes, or methodologies. Dell Technologies believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." DELL TECHNOLOGIES MAKES NO RESPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Use, copying and distribution of any Dell Technologies software described in this publication requires an applicable software license. © 2023 Dell Inc. or its subsidiaries. All Rights Reserved. Dell and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.