Designing Multi-Tier Applications Using Azure App Services

Parag Bhardwaj

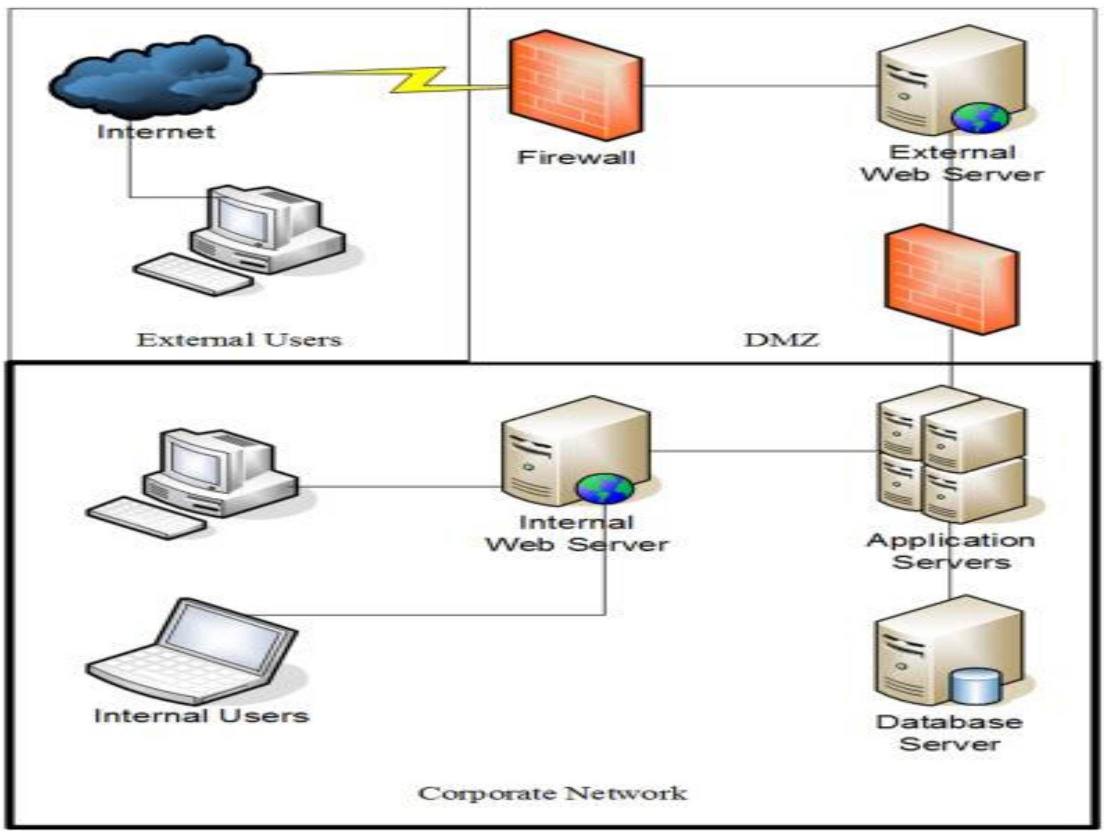
Abstract

Building and deploying cloud based applications with Azure App Services provides a robust and scalable option for designing multi-tier applications. Azure App Services is a fully managed platform that makes it easier for developers to create, deploy, and host web apps, mobile backbends, and RESTful APIs—with minimal infrastructure management. With Azure's integrated tools and services, this abstract will explore how to benefit and methodology for designing multi-tier applications. Once again, in a typical multi-tier architecture, the application is separated into different layers like presentation, business logic and data layers, and is hosted independently, in order to improve scalability and security. This separation Azure App Services makes possible indulging in different hosting environments – such as Web Apps for frontend layers, Azure Functions for backend logic and Azure SQL Database or Cosmos DB for the data layer. They make seamless integration between layers of these services possible. The research focuses on identifying some key components of a multi-tier design that utilizes Azure App Services for load balancing, auto scaling, and high avail- ability. It also looks at security practices like API management, identity authentication and network isolation to protect data from compromise and compliance with regulatory norms. The paper further discusses best practices of deploying, monitoring and managing multi-tier applications deployed in the cloud using Azure DevOps tools and using Continuous integration and continuous deployment (CI/CD) pipelines.

Introduction

The designing of multi-tier applications is a common approach to software development today where an application is divided into layers, or tiers, each performing one set of functions like presentation, business logic, and data management. This architecture increases scalability, maintainability and security making it suitable for cloud based applications. Microsoft Azure provides a set of services and tools to build applications which Azure App Services provides fully managed platform. Developers will be able to deploy web apps, mobile backends, and APIs using Azure App Services and the overhead of infrastructure management is offloaded to the cloud. The advantage gained is that businesses can concentrate on their application logic development, with no concerns about underlying hardware or operating system management.

In a multi-tier application architecture the frontend (presentation layer), backend (business logic layer) and data layer is often placed separately so that the frontend can be easily changed (without changing the existing infrastructure), the backend can scale horizontally based on the need, and the data is securely hosted in a separate place. This separation is supported by Azure App Services, which provides components particular for each layer.



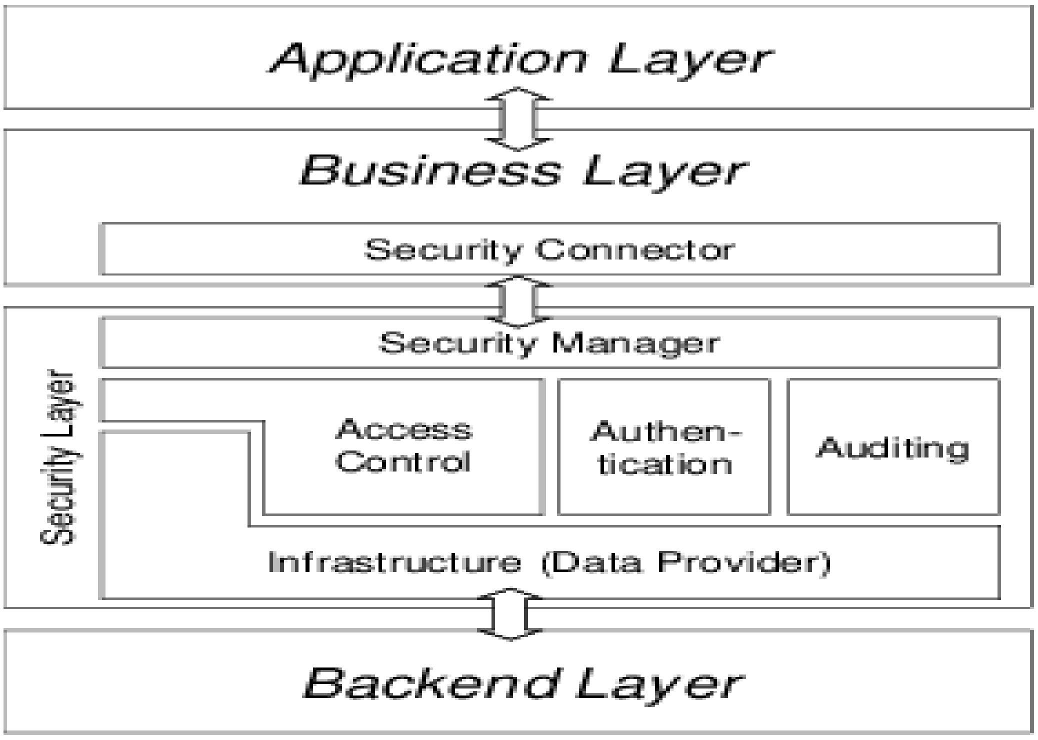
The presentation layer is hosted on the Web Apps feature and business logic and background processing are managed with the help of Azure Functions or App Service plans. Azure gives a lot of options for the data layer, with managed databases like Azure SQL Database or Cosmos DB which offer high availability, scalability and security. This cloud native approach does not only support horizontal and vertical scaling, but simplifies application management with task automation, such as load balancing, security updates and performance monitoring. To develop secure, efficient and scalable services for multi tier applications built on Azure, services for authentication (Azure Active Directory), secret management (Azure Key Vault) and handling APIs (API Management) can integrate together. In this paper, we delve into the architecture, components, and best practices to design for, and deploy, multi-tier applications on Azure App Services as a blueprint for developers to construct cloud native applications that are both future proof and strong.

Rationale of the study

Azure App Services provides a way to design multi-tier applications which are scalable, secured and have high performance in the fast changing digital world, hence, the study of design of multi-tier applications using Azure App Services is warranted. With the proliferation of cloud migrations when businesses migrate their applications to the cloud, there is a dramatic need to have architectural approaches that are flexible, reliable, and easy to manage. The use of multi-tier architecture, wherein the applications are broken down in different layers presentation, business logic and data allows in better performance, maintaining and security. Such architectures are well served by Azure App Services which is a fully managed platform offering a powerful and efficient platform to implement such architectures, with features like automatic scaling, load balancing, and integrated security services that make development easy and reduce operational overhead. In this study I investigate benefits on the side of using Azure App Services as a host for multi-tier applications including both technical and business angle. Azure can help developers by taking on the sweat details of the underlying infrastructure, and by providing a deep tool set of tools at developer's disposal. Faster time to market, higher scalability and improved security are just some of the benefits business can gain by adopting this approach. Moreover, with the industry shifting towards micro services and cloud native applications more and more, being able to design multi-tier systems using Azure App Services is essential for developers and organization who want to be competitive in the cloud ecosystem. This study looks to provide actionable findings and guidance on how to use Azure's platform to build reliable, scalable, and secure multi-tier applications which satisfy the requirements of modern business needs.

Definition and Benefits of Multi-Tier Architecture

M ulti tier architecture (also called n tier architecture) is a design in which an application is structured around the concept that it includes several layers, each n with its own particular functionality separated from the others. Typically, multi-tier architectures consist of three primary layers: Is built for 3 layers: the presentation layer, the business logic layer, the data layer. The presentation layer is where the user sits, which is to say that it's where the UI and UX live, where the information is presented and where the inputs come in. The core functionality of the application, processing data, applying business rule and computing can be found in this layer. In the end, the data layer deals with storing, fetching, and altering data (usually in a database, but other data storage systems could be utilized as well).



There are plentiful benefits of the use of multi-tier architecture, most notably for cloud based applications. In the beginning, it provides scalability by making each tier scale as workload demands require. For example, if the business logic or date processing want to handle higher loads, just one tier can be scaled and not the entire application as a whole. Second, It makes it easier to maintain because changes occur in isolated layers. For example, it allows us to make changes to the user interface without impacting the business logic or data storage layers and thus make this subsystem less complicated and more long term maintainable.

Third, allowing data to be isolated in the data layer with a multi-tier architecture makes it easier to implement access controls and encrypt the data. Moreover, it makes things flexible (you can even use different technologies for each layer, for example, different databases for data storage and different frameworks for front end). Additionally, this method allows for simpler integration with other service or platform, thus being easier to work alongside other services in the future.

Key Components

Presentation Layer

Among the four layers of the multi–tier architecture the presentation layer is the upper most layer also known as the user interface (UI) layer. It is in charge of communicating with the user in the most direct way possible by showing business logic information and allowing the user to submit commands or information. This layer is used for displaying the visual part of an application, or buttons, forms, images, text fields and navigation elements of the application. It is the medium and means by which users talk to the system.

The user friendliness is ensured in the presentation layer, but is also expected to offer an intuitive and responsive interface to deliver a good user experience (UX). Web applications are often built off of web technologies including HTML, CSS, and JavaScript (or a mobile development framework) for mobile applications. Validation of the data before it is passed to the business logic layer is one of its key responsibility. All the examples I can find validate user inputs in the presentation layer to make sure these inputs satisfy the required format or constraints.

In cloud based multi-tier applications the presentation layer in most cases are hosted on Azure App Services or AWS Elastic beanstalk which takes care of deployment and scaling of web applications. Moreover, frameworks such as React, Angular or Vue.js can be utilized in modern applications for creating dynamic and responsive interfaces. This layer is crucial to separation of concerns by providing a means for the business logic and data management to change without changing how the data is presented to the user.

Business Logic Layer

The application layer, or middle layer, is the business logic layer of a multi-tier application. This forms the core layer that defines the core logic of how the business rules and prerequisites operate on data. It is in charge of calculating, transforming data, validating etc., and also, forwarding data from presentation layer to data layer.

In the next layer, business rules are implemented on top of this to complete the needs of the business domain or application. For example, taking a case of an e-commerce platform as an application, the business logic layer will handle functions such as calculating total prices, processing orders, applying discounts, and managing levels of inventory. Alternatively, this layer can more easily handle more complex workflows, like handling payment processing or sending email notifications or customer support integrations with external services or APIs.

The business logic layer, which sits between a front end and a back end data layer, handles it. Intake is done from presentation layer, data is processed according to the predefined business rules and result feeded to data layer or presentation layer. Normally, it is accomplished through the use of server side programming languages, like Java, Python, C#, or, Node.js.

However, in the case of cloud environments, the business logic layer can be implemented when using services such as Azure Functions or AWS Lambda which allow developers to deploy microservices or serverless functions which automatically scale up or down against demand. Business logic layer is crucial to making sure that business operational is efficient, consistent and secure.

Data Layer

A data layer, also called a persistence layer, is the lowest tier in a multi-tier architecture, and handles data storage and retrieval and manipulation. The business logic layer calls on this layer to store and retrieve data from the databases, file systems, or any other data storage that the business uses. It makes data durable, consistent and accessible over the application.

The data layer often stores the data in a relational database like SQL Server, MySQL or PostgreSQL, or a NoSQL database like MongoDB or Cassandra, depending on whether the data is or the requirements of the application. This layer includes the tasks like data integrity, transaction and data security. It also deals with complex queries, and optimized data retrieval to guarantee performance regardless of the level of load.

The data layer is one area where we have abstracted away the complexity of how data is stored from the world of business logic and presentation. So for example the business logic layer doesn't need to concern itself with what or where the data is stored; it just goes through an API or set of predefined data access methods. This makes sure that separation of concerns will be achieved (again, a core multi-tier architecture principle).

In cloud based systems, managed databse services such as Azure SQL Database, Amazon RDS, Google Cloud Firestore may be included in the data layer, given the high availability, backup, and automatic scaling provided by them. By reducing the administrative burden on developers, these services give a means for securely, dependably and scalably storing data. The data layer is important in allowing an application to process huge amounts of data in a very efficient and high performance nature, while maintaining security.

Literature Review

Grozev, N., & Buyya, R. (2015). To capture the performance modeling and simulation of three tier applications in cloud and multi cloud environment, we need to build models to predict and optimize the performance of these applications across different layers, which include presentation, business logic, and data storage. For these models, we consider cloud and multi cloud environments and consider the impact of resource allocation, network latency, load balancing, and scalability on overall system performance. Through simulating workloads, traffic patterns and resource demands under different conditions, bottlenecks, the potential for failure, as well as areas for optimization are identified. With multiple cloud providers supporting multiple different infrastructure capabilities the model has to encapsulate this complexity for optimal resource distribution and failover strategies. Simulations help make decisions regarding cloud resource scaling, redundancy and improvement of the network, monitoring closely performance metrics like response time, throughput, system reliability, and so on. Performance modeling and simulation techniques can be used to fine tune a business's three tier applications on moving towards efficiency, cost cuts and better user experience in dynamic cloud environments.

Hajjat, M., et al (2015). The problem of measuring and characterizing the performance of interactive multi-tier cloud applications requires (i) measuring and characterizing the responsiveness of applications that span different layers (such as the user interface, business logic, and data storage); (ii) measuring and characterizing the scalability of these layer interactions; (iii) measuring and characterizing their reliability. Response time, throughput, latency, and system availability are key performance metrics for interactive applications involving real time user interaction. Performance characterization, by studying the interaction of different tiers of application at various loads and under different network conditions. For instance, you simulate user traffic through load testing, stress testing, and monitoring in each tier to identify performance bottlenecks. Data flow is optimized such that delayed occurs between tiers as infrequently as possible, especially in a distributed cloud environment, where latency plays a critical role in user experience. To verify that the application will experience no performance degradation when dealing with different workloads, scalability tests are run. Using these metrics, developers can adjust resource allocation, refine database queries, and enhance the whole efficiency of the interactive multi — tier cloud applications.

Khasnabish, J. N., et al (2015). In multi-tier cloud systems tier centric resource allocation tries to answer the following question: how can resources be allocated across different application tiers (presentation, business logic, and data) in order to maximize the exploitation of their specific needs and workload demands. Often, resources needed by each tier will differ from each other: the presentation tier could require high network bandwidth and low latency, while the data tier could require storage and database performance on steroids. Cloud systems can dynamically allocate resources by using tier centric strategies; each tier will be provisioned right compute, right storage and the right network to achieve optimum performance. Techniques such as autoscaling, load balancing, and resource prioritization ensure that resources are fed across a tier in an efficient manner. With this approach, users can manage their varying workloads without worrying about wasted resources, and also prevent bottlenecks. The abstraction leverages resource (sharing) parameters within each tier in order to align resource allocation to the specific requirements of each tier, thereby improving performance and cost-efficiency of multi-tier cloud applications.

Han, R., et al (2014). Cost aware and adaptive elasticity in multi-tier cloud applications is the ability to dynamically adjust resources to real time demand in a cost efficient manner. In the cloud, elasticity means that the resources can automatically grow or shrink, as needed for the workload, thus providing the best performance for the application, without over provisioning or underutilization. In multi-tier applications, different tiers (presentation, business logic and data) may have various scaling requirements. For example, even though the data tier is likely to need some scaling, it has nowhere near the scaling demands of the presentation layer — where response time must keep up with rapidly changing user traffic. A cost aware approach monitors resource usage (CPU, memory and network bandwidth) in real time in order to make decisions that trade performance for cost. Predictive analytics and auto-scaling policies that let you see the demand ahead of time and respond proactively when resources are about to become scarcer or when bottlenecks are expected are called adaptive elasticity.

With the help of these strategies, organizations can ensure that their multi-tier cloud applications are not only responsive but also cost effective, without creating overwhelming quantity of waste at the cost of responsiveness.

Huang, D., He, B., et al (2014). Resource management in multi-tier web application is then surveyed with techniques for allocating and optimizing resources across presentation, business logic and data tiers. Different resources are needed for each tier including compute power for processing logic, storage for data management, as well as network capacity for handling user requests. Resource utilization in multi-tier application result in managing resources over resources utilization taking place, resulting in scalability, performance and availability. Load balancing is one approach for distributing traffic twenty to distribute evenly across servers so that no single tier is overloaded.caching mechanisms cache frequently accessed data to reduce gotchas on database, and autoscaling is used to dynamically adjust resources according to demand, thus particularly responding to fluctuating workloads.h tier requires distinct resources, such as compute power for processing logic, storage for data management, and network capacity for handling user requests. Effective resource management in multi-tier applications ensures scalability, performance, and high availability.

Addya, S. K., et al (2021). Strategic combination of VMs over multiple cloud platforms leading to VM coalition for multi-tier applications over multi cloud environments aims to optimize the performance, cost and availability of the application. In this multi cloud setup, different cloud provider have their own pricing, resources and geographic spread to the place. Businesses could use VM coalitions allocating VMs across multiple cloud providers for each tier of a multi-tier application based on its specific needs. For instance, a presentation tier could be hosted on a cloud provider having better latency and global presence, so as to place the data tier on a provider providing better storage and database services. VM coalition selects the most suitable cloud environment for each workload, dynamic scaling allows, fault tolerance, making optimized resource utilization. It also avoids vendor lock in and keeps high availability through redundancy across the clouds. Improved cost efficient, flexibility, and resilience is achieved through coalition of VMs across multiple clouds on multi-tier applications.

Wilder, B. (2012). Microsoft Azure gives a robust structure for building cloud architecture patterns of scalable, reliable, and cost efficient cloud solutions. Examples of key patterns are the Microservices pattern where applications are split into small, relatively loosely coupled services which can run independently and can be scaled independently. These microservices are often deployed by using Azure Kubernetes Service (AKS) and Azure Functions. Their Serverless pattern uses Azure Functions to let developers write code without managing infrastructure, scaling to handle demand automatically and minimizing costs by paying only for the time their code executes. The Event-driven pattern, exposed by Azure Event Grid and Azure Service Bus, makes components decoupled, kicks off actions when events happen, such as data changing or user interaction. Azure App Services still employs the Monolithic pattern but for smaller applications it provides simple deployment and scaling. With Azure, your workloads can be distributed wherever they are most cost effective and meet regulatory requirements, while still giving you flexibility and compliance with Hybrid cloud capabilities. These patterns help businesses achieve performance, scalability, and cost efficiency improvement.

Security Features in Multi-Tier Applications

Designing multi-tier applications demands security as many businesses are migrating operations to cloud. Well, a multi-tier architecture by itself can act as an added security as different functional areas of the application can be split into various layers and each layer would also have its security options. When a multi-tier application is well designed the security layer for each layer can be segregated and it should not affect the rest of the system if there is vulnerability in one of the layer. Below are the key security features to consider when designing multi-tier applications:

1. **Authentication and Authorization (Identity Management)**

Authentication verifies the identity of users or systems accessing the application, while authorization determines whether a user or system has permission to perform specific actions or access particular resources. In multi-tier applications, Azure Active Directory (AAD) or other identity management solutions are often used to authenticate users across all layers. Role-based access control (RBAC) can be implemented to ensure that only authorized users can access sensitive data or perform critical business operations. The authentication process is usually handled at the presentation layer, while authorization is managed at the business logic layer, ensuring proper access control for each user based on their roles.

2. **Encryption**

Data is encrypted (both in transit and in rest) to protect it. Transport Layer Security (TLS) protects data in transit by encrypting data transmitted over the network during the client to server and application to application communications. For data in rest, you can use Azure Key Vault or similar service to securely store and manage the sensitive data like passwords or API keys. Encryption of database can also be deployed to secure the data stored in data layer that ensures the compliance of data protection to the industry standards.

3. **API Security and Secure Communication**

In a multi-tier application, often APIs are invoiked to delegate the interaction between the frontend, business logic, and data layer. One thing that’s extremely important is API security because when you expose your APIs for any type of service, it’s important that you allow only authorized users to access it. To secure API endpoints, techniques such as OAuth 2.0, API keys, and JWT (JSON Web Tokens) are most commonly used.

4. **Network Security and Segmentation**

Network security is essential in ensuring that data flows securely between the different layers of a multi-tier application. By using **Virtual Networks (VNets)** in cloud environments like Azure, network traffic can be isolated and controlled. For example, access to the data layer can be restricted to the business logic layer through private subnets, ensuring that only authorized components of the application can communicate with sensitive databases. Network security groups (NSGs) and **firewall**

rules can be used to define access controls between different tiers, ensuring that the presentation layer and external networks do not have direct access to the database.

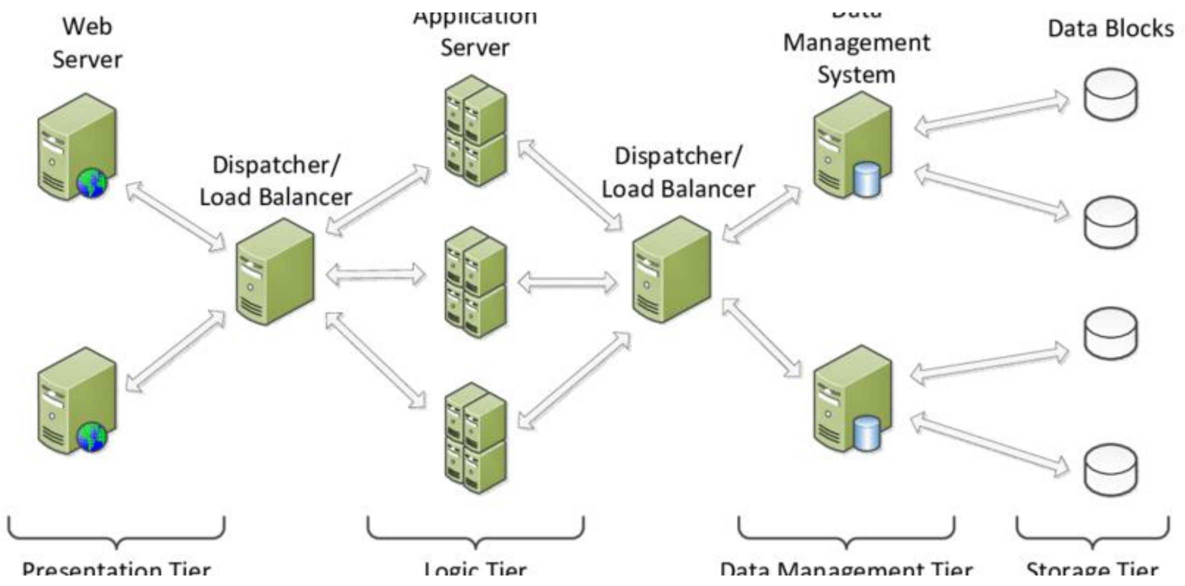
5. **Monitoring and Auditing**

Continuous monitoring and auditing are vital for identifying and responding to security incidents. Security tools like Azure Security Center, Azure Sentinel, and third-party solutions can be used to monitor the application in real-time for suspicious activities or vulnerabilities. Logs generated by the different layers of the application can be aggregated and analyzed to detect abnormal patterns, unauthorized access attempts, or other security breaches. These logs also play an essential role in auditing user actions and maintaining compliance with regulatory standards.

By leveraging these security features, multi-tier applications can ensure the protection of sensitive data, secure access control, and compliance with security standards, while maintaining a separation of concerns between the different application layers. This layered approach to security makes it more difficult for attackers to exploit weaknesses in one area and gain access to the entire system, thus enhancing the overall resilience of the application

Challenges in Multi-Tier Application Design and Deployment

While such advantages carry many economic values, they also bring with them a set of challenges that must be dealt with explicitly when implementing successfully. Complexity in system architecture is one of the largest challenges. Such type of applications by definition are multi layered: layers have different responsibilities, which lead to different requirements. It makes the application harder to design, configure and maintain. Careful planning and the right selection of the technologies are needed, to ensure efficient and secure communication between layers. Like most other computer programs, databases can be hampered by bad design choices, e.g. wrong tier boundaries, or inadequate layer integration.



Another challenge is multi-tier application scalability and resource management. Different layers of the application have different scalability needs, and hence managing these resources is difficult. For example, the business logic layer may need more computational power when the business is running at peak usage and the data layer may need storage. In a cloud based environment, auto scaling and load balancing is frequently employed to deal with fluctuating loads but such mechanisms have to be sensitively configured so as not to overspend or underspend on resources. Moreover, this can result into the over scaling of some tiers and more cost as well as inefficient use of cloud resources.

There are also big challenges for security management in multi-tier applications. Separating responsibilities into individual layers is an effective mechanism to prevent the proliferation of sensitive data, but comes at a cost of extra complexity in controlling access to the separate layers. Different mechanisms, for instance encryption, secure APIs, and good authentication systems, are implamented to ensure that each layer is appropriately secured. But if you manage these security layers in a consistent way across every tier, that’s resource intensive and likely error prone, so that risk of vulnerabilities is high. Deploying and integrating multi-tier applications on different environment may result in deployment and integration issues, most especially in the hybrid cloud or the multi cloud. It’s cumbersome to manage pipelines for continuous integration and continuous deployment (CI/CD), maintaining compatibility across layers, and testing end to end functionality. In addition, the integrity of the application has to be retained in face of distributed layers data consistency and transaction processing. Complex multi-tier deployments render the need for tools for efficient monitoring, troubleshooting and debugging even more critical.

Results and Discussion

Feature/Service	**Azure App Services**	**Traditional Web Hosting**	**Benefits of Azure App Services**
Scalability	Auto-scaling, manual scaling, horizontal scaling	Limited or no scaling options	Scalable with minimal configuration, automatically adjusts to traffic demands.
Infrastructure Management	Fully managed, no infrastructure management required	Requires manual setup and management	No need to manage servers, reducing operational overhead.
Performance Monitoring	Integrated with Azure Monitor and Application Insights	Limited monitoring or external tools	Built-in monitoring and real-time performance tracking for optimization.
Security	Managed identity, API management, SSL certificates, Azure Key Vault	Requires manual configuration for security	Advanced security features such as role-based access control (RBAC), encrypted data storage, and secure APIs.

Cost Efficiency	Pay-as-you-go, pricing tiers based on usage	Fixed costs, resource over-provisioning	Flexible pricing, reducing costs during low-usage periods.
Integration with Other Azure Services	Seamless integration with Azure SQL, Cosmos DB, Azure Functions, etc.	Requires manual integration with third-party tools	Easy integration across multiple Azure services for enhanced functionality.
Deployment Ease	Continuous Integration/Continuous Deployment (CI/CD) via Azure DevOps	Manual deployments and complex processes	Streamlined CI/CD processes to automate testing and deployment cycles.
Data Management	Azure SQL Database, Cosmos DB, Blob Storage	Limited database options, third-party databases	High-performance, globally distributed data storage options.
Customization	Full control over app settings, scalable plans	Limited customization and configurations	More flexibility in resource scaling and application setup.

The table shows a comparison of the key features and services between Azure App Services and Traditional Web Hosting, and explains why by default Azure App Services is the right platform choice for deploying a multi tier application. One of the biggest advantages of Azure App Services is the ability to auto-scale, manually scale, or perform horizontal scaling. With this, applications can scale with traffic demands without manual intervention, whereas traffic demands that are typically unscalable, or require complex configuration, for traditional web hosting.

Azure App Services handle all the Infrastructure Management for you, meaning there's no need for manual setup and server management that is typical of most traditional hosting solutions. This saves on operational overhead so that time can be spent by the developers on the application rather than the infrastructure.

In order to monitor performance, Azure App Services connects to Azure Monitor and Application Insights for real time activity which is monitored and continuous optimization can be done. Traditionally hosted sites often have minimal internal, or external, tools to monitor performance and make tuning difficult. Azure App services provide a high level of security with robust security such as managed identity, API management, SSL certificates and integration with Azure Key Vault that is much higher than what is required for a traditional hosting. Cost effective, Azure App Services are charged per use basis whereas traditional hosting follow a fixed cost structure. Moreover, integration with other Azure services like Azure SQL, Cosmos db and Azure Functions is seamless and one has a smoother integrated and efficient ecosystem to work with than a traditional solution. deploying your Azure App Services is easy via CI/CD pipelines which are provided by using with Azure DevOps, making updating simple and easy on deployment.

Conclusion

Finally, we can say that designing multi-tier applications with Azure App Services is a very powerful, flexible and efficient solution for application designing and deployment. The advantage of Azure App Services is that it provides a fully managed environment which takes care of the important stuff such as scalability, infrastructure management, performance monitoring and security and allowing the developer to concentrate more on application functionality and less on server management. Stay9 also integrates interesting features like auto-scaling, continuous deployment and easy to use integration with other Azure services like databases, functions and API management. Built in monitoring tools and the ability to scale individually each layer of it multi-tier application allows for applications to adapt to variable traffic and performance demands without sacrificing availability. Security features like managed identity, encryption and role based access control (RBAC) additionally make sure that sensitive data and processes are secured across each layers. Furthermore, the pay as you go model makes the finances more cost efficient, enabling the cost of operation to be optimized while still running at optimum levels. In the end, Azure App Services makes it easy to design, deploy and manage multi-tier applications, providing organizations the scalability, flexibility and security that today's cloud based solutions require. This means it is a fantastic option for those who want to develop resilient, robust and scalable applications, as well as to avoid managing infrastructure overheads.

References

1. Grozev, N., & Buyya, R. (2015). Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments. *The Computer Journal*, *58*(1), 1-22.
2. Hajjat, M., Pn, S., Sivakumar, A., & Rao, S. (2015, April). Measuring and characterizing the performance of interactive multi-tier cloud applications. In *The 21st IEEE International Workshop on Local and Metropolitan Area Networks* (pp. 1-6). IEEE.
3. Khasnabish, J. N., Mithani, M. F., & Rao, S. (2015). Tier-centric resource allocation in multi-tier cloud systems. *IEEE transactions on cloud computing*, *5*(3), 576-589.
4. Han, R., Ghanem, M. M., Guo, L., Guo, Y., & Osmond, M. (2014). Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, *32*, 82-98.
5. Huang, D., He, B., & Miao, C. (2014). A survey of resource management in multi-tier web applications. *IEEE Communications Surveys & Tutorials*, *16*(3), 1574-1590.
6. Addya, S. K., Satpathy, A., Ghosh, B. C., Chakraborty, S., Ghosh, S. K., & Das, S. K. (2021). CoMCLOUD: Virtual machine coalition for multi-tier applications over multi-cloud environments. *IEEE Transactions on Cloud Computing*, *11*(1), 956-970.
7. Wilder, B. (2012). *Cloud architecture patterns: using microsoft azure*. " O'Reilly Media, Inc.".
8. Moreno-Vozmediano, R., Montero, R. S., Huedo, E., & Llorente, I. M. (2018). Orchestrating the deployment of high availability services on multi-zone and multi-cloud scenarios. *Journal of Grid Computing*, *16*, 39-53.
9. RahimiZadeh, K., AnaLoui, M., Kabiri, P., & Javadi, B. (2015). Performance modeling and analysis of virtualized multi-tier applications under dynamic workloads. *Journal of Network and Computer Applications*, *56*, 166-187.
10. Oosthuizen, A. D. (2014). Developing a multi-tenant, media-marketplace architecture with Microsoft Azure and ASP .NET.

11. Papaioannou, A., & Magoutis, K. (2013, December). An architecture for evaluating distributed application deployments in multi-clouds. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science* (Vol. 1, pp. 547-554). IEEE.
12. 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.
13. Bahga, A., & Madisetti, V. K. (2013). Rapid prototyping of multitier cloud-based services and systems. *Computer*, *46*(11), 76-83.
14. Fylaktopoulos, G., Goumas, G., Skolarikis, M., Sotiropoulos, A., Athanasiadis, D., & Maglogiannis, I. (2015). CIRANO: An integrated programming environment for multi-tier cloud based applications. *Procedia Computer Science*, *68*, 42-52.
15. Chi, R., Qian, Z., & Lu, S. (2011, June). A heuristic approach for scalability of multi-tiers web application in clouds. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (pp. 28-35). IEEE.
16. Sanaei, Z., Abolfazli, S., Gani, A., & Shiraz, M. (2012, August). SAMI: Service-based arbitrated multi-tier infrastructure for Mobile Cloud Computing. In *2012 1st IEEE International Conference on Communications in China Workshops (ICCC)* (pp. 14-19). IEEE.
17. Sonmez, C., Ozgovde, A., & Ersoy, C. (2017, May). Performance evaluation of single-tier and two-tier cloudlet assisted applications. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 302-307). IEEE.
18. Suleiman, H. (2019). Service-Level-Driven Load Scheduling and Balancing in Multi-Tier Cloud Computing.
19. Koziolek, H. (2011, June). The sposad architectural style for multi-tenant software applications. In *2011 Ninth Working IEEE/IFIP Conference on Software Architecture* (pp. 320-327). IEEE.
20. Babu, S. S., Sarma, C. S., Vijaylakshmi, Y., Kalyankar, N. V., Buyya, R., Yeo, C. S., & Venugopal, S. (2012). Scalability of multi-tier transactions towards data confidentiality for cloud applications. *International Journal of Soft Computing and Engineering*, *2*, 247-250.