Scientific Publications

Journal of Artificial Intelligence and Big Data, 2022, 2, 1207
www.scipublications.org/journal/index.php/jaibd
DOI: 10.31586/jaibd.2022.1207

Review Article

Building Scalable and Secure Cloud Architectures: Multi-Region Deployments, Auto Scaling, and Traffic Management in Azure and AWS for Microservices

Manogna Dolu Surabhi *

Quality Assurance Analyst, General Motors, Michigan, USA

*Correspondence: Manogna Dolu Surabhi (manognadolu@outlook.com)

Abstract: The last few years have seen an increased adoption of cloud infrastructure, which has in turn led to a growth in large-scale distributed architectures in data centers to accommodate cloud resource elasticity and resiliency better. Selecting the right approach to build secure, scalable, and reliable cloud infrastructure within a budget is always a challenge. This text focuses on offering practical solutions for designing and building a secure, scalable, and reliable cloud-based infrastructure where auto-scaling and multi-region deployments are the two key approaches to offer high availability. It covers designing secure and scalable microservices using cloud platforms. The content will provide an understanding of public cloud architecture, the design of microservices running on the cloud, and also the design patterns used in the cloud era. With real-world examples, you will learn how microservices can enable scalable distributed systems. Furthermore, you will be walked through multi-region deployments, auto-scaling, and traffic management in cloud environments, using a sample environment setup and useful tips and tricks for monitoring. Finally, you will see a mock implementation of cloud infrastructure on-premise for a private cloud or single-node cloud. By the end of this text, you will be able to build, manage, and deploy a highly scalable and reliable cloud-ready solution [1].

Keywords: Cloud Infrastructure, Distributed Architectures, Auto-scaling, Multi-region Deployments, Microservices, High Availability, Cloud Security, Resource Elasticity, Traffic Management, Private Cloud

How to cite this paper:
Surabhi, M. D. (2022). Building Scalable and Secure Cloud Architectures: Multi-Region Deployments, Auto Scaling, and Traffic Management in Azure and AWS for Microservices. *Journal of Artificial Intelligence and Big Data, 2*(1), 127–140. Retrieved from https://www.scipublications.com/journal/index.php/jaibd/article/view/1207

Received: August 3, 2022
Revised: October 2, 2022
Accepted: December 18, 2022
Published: December 27, 2022

1. Introduction

Cloud computing has evolved significantly over the past decade to offer various levels of services that have, in turn, been adopted by different types of customers to cater to their existing and emerging hardware and software requirements. Some of these varied services include Infrastructure as a Service, Platform as a Service, and Software as a Service. There have been significant repository changes recently in the Big Data space that have added new services that can be used. These could be looked at as In-Service and After-Service using the letters 'S' or 'A' instead of 'P'.

Security is a major concern and ever-evolving feature of cloud systems. As developers incorporate more features to balance scalability issues related to the demand for cloud systems, security and privacy concerns have stagnated. Therefore, the objective of this project-based course is to familiarize students with the design, integration, testing, and deployment of scalable, performant, and secure cloud systems. Students will explore security in cloud systems and will learn to implement auto-scaling, vehicle-to-cloud traffic management, and multi-region cloud deployments. Students will focus on core skills of high technical value, including design, integration, testing, and deployment of cloud systems using hands-on experience and tools.

1.1. Background and Significance

Scalable traffic management has become an indispensable vehicle for continuous cloud infrastructures, such as modern microservices-based applications that can recover from server issues. Scalable and fast traffic reroutes are crucial in such elastic computing environments in reactive engineering practices by having innovative approaches to facilitate microservices. Currently, the methods used to satisfy global traffic management requirements for web applications point to a single deployment. This approach necessarily falls under traffic management as automatic failover and fast server responses for what they need may be significant or delayed since it is based on the moment of urgent conditions. This singular within-zone setup also assumes that the system can handle all the capacity that the deployment will ever need to use [2].

1.2. Research Objectives

The primary function of this research is to provide IT professionals and architects with practical insights into high-impact practices to optimize and secure their multi-region deployments with microservices in public clouds. The specific research objectives are as follows. Overall Research Objective: To develop and maintain working multi-region microservice deployments, addressing the inherent complexities that cloud services are exposed to when organizations are considering or have already extended their reach using more than one geographical region to deliver such services. Objective 1: Examine cloud services scalability and availability considerations together with regional geolocation complexities of such distributed geographies. Objective 2: Develop a working multi-region deployment of containerized metaservices in public clouds. Objective 3: Research and validate automated traffic management strategies that can ration and route user requests among different cloud providers. Objective 4: Showcase through user scenarios how to securely deploy stateless web front-end technologies in asynchronous development to leverage caching to deliver low latency in serving user requests around the globe using multi-region deployments running through multiple cloud providers.

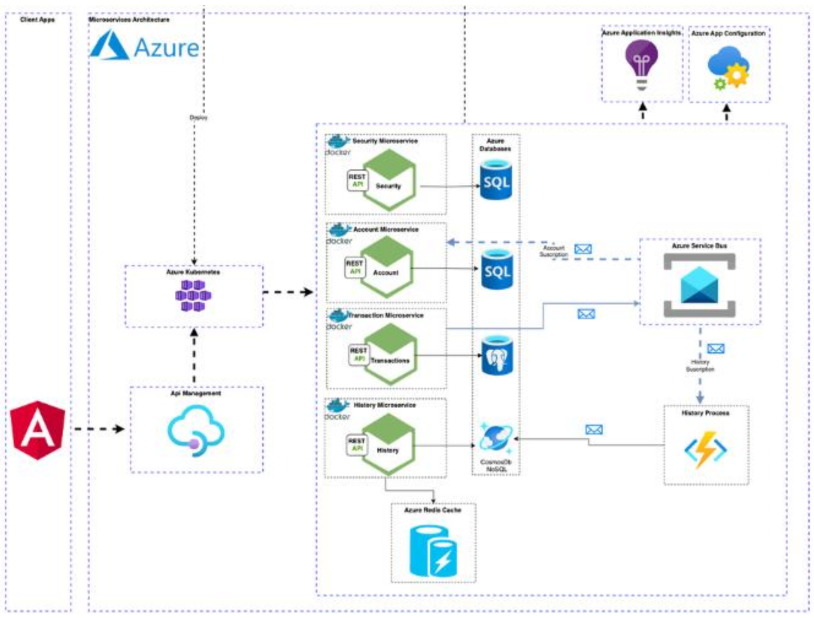

Figure 1. Building Scalable Microservices Leveraging the Power of Azure Cloud Services

1.3. Scope and Limitations

We discuss the provisioning and orchestration of a cloud. We feature microservices and their communication in a tightly controlled way. Microservices are currently often in predefined containers or virtual machines, which in turn may be managed in a container service. This layer is not challenged. What is needed and discussed in this text is the infrastructure to support cloud deployment, including networking, DevOps, and SecOps processes, monitoring and logging, virtual networking, security, and management. We focus on the platform for the creation and supervision of VM- or container-based services in the cloud. Relevant topics are LAN, WAN, and other communication paths, and the possibilities by which these different applications can (and will) be interconnected. The reason for this text is that the security of clouds is difficult in a networked world, and that 'best practices' often do not help in state-of-the-art applications [3].

This text will come out in concurrent versions for different cloud systems, making a comparison easy. For both systems, there are differences, but the main concepts are uniform. This text bridges the gap by sharing the design and integration similarities. Such similar implementations are more robust and require, for instance, only a single, shared programming layer and overhead. The base definitions are similar, and a cloud user should be able to jump from one to another with only a little bit of training. There will be situations in the cloud in which a certain service does not exist or is not publishable. We will give an alert if such a platform-specific service becomes critical. The structure secured by regional deployment and traffic management will in turn support the automatic and semi-automatic rules imposed for scaling in the cloud of microservices. With security constraints applied to these scaling rules, we achieve that microservices communication details will be explicit and known in advance, during design, and never be part of an emergent architecture [4].

2. Cloud Architecture Fundamentals

Enterprises and small businesses are moving industry verticals into the top public cloud providers. Business values related to time to market, fast scalability, enhanced security, redundancy, and application lifecycle management, as well as the increase of industry-specific cloud services and APIs, make this transition an easy decision. The holistic approach to crafting cloud business applications and their architectures, discovering sufficient flexibility and resiliency based on the application's requirements, is technically challenging for leadership and architects. Most experienced software and workload engineers and architects familiar with cloud platforms and development ideally collaborate and concur on the canonical architectures of their creations. Enterprises frequently lack sufficient experienced resources, and small businesses are resource-constrained. For these businesses, end-to-end architecture can be left up to inexperienced consultants or developers who are not in an optimal position to design for ongoing operations [5].

Guidelines are needed to assist architects in meeting the main business and technical challenges outlined here. Such guidance is necessary not only to facilitate development but also to result in fast, secure, adaptable, and scalable cloud architectures for today's business goals, ensuring that once deployed and tested, they remain in this state over time. This discretion in providing guidelines for the creation of well-formed applications facilitates the execution of a set of repeatable, well-engineered procedures demonstrating scalability and resiliency, leading to a holistic understanding of concepts. Many times, scalable cloud architectures share fundamental building blocks. In this work, we craft scalable and regionless cloud architectures for web services that quantify well within desired business tolerances throughout their lifetime, with only a limited set of architectural primitives. A set of clear architectural guidelines results, including scalable web services microarchitecture, load balancing, auto-scaling, health monitoring, service guarantees, information technology resilience, leveraging multi-region deployment, and information lifecycle management.

2.1. Definition and Components

Containers – decoupling and isolating application components – are key, with each component being provided 'just enough' to compute and horizontally scalable. Each application runtime and library set is baked into a container image suitable for deploying on a container orchestrator and is tagged with a version number. Endpoints to microservices can be exposed only where necessary, and where possible, their identity and access are through structured messages against authentication and authorization using policies associated with the applications' identity. Containers package up microservices inside one or more presentation layers and define what each microservice can do through the services it presents – endpoints providing external services, background jobs, queue consumers, and so on. These coarser-grained components should not have dependencies across them, and architects should be working towards independence where it does not simply happen. Such dependence is a sure sign of architectural error. By dividing services to the point that dependence can be eliminated, developing the component is governed – the autonomy maximized – while delaying and simplifying any dependencies ensures that it can be managed by the team without becoming a monstrous interface [6].

Equation 1: Multi-Region Deployment

Latency Equation:

$$\text{Latency}_{\text{total}} - \text{Latency}_{\text{network}} + \text{Latency}_{\text{processing}}$$

Where:

$$\text{Latency}_{\text{network}} - \text{Distance}_{\text{regions}} / \text{Speed}_{\text{light}}$$

$$\text{Latency}_{\text{processing}} - \text{Processing Time}_{\text{service}}$$

Data Consistency Model:

$$\text{Consistency}_{\text{model}} - (\text{Strong, Eventual, Causal})$$

Availability Calculation:

$$\text{Availability} = 1 - \prod_{i=1}^{n} \left(1 - \text{Uptime}_{\text{region}_i}\right)$$

Where n is the number of regions.

2.2. Benefits and Challenges

2.2.1. Benefits of Multi-Region Deployments

Deploying microservices in geographically distributed data centers comprising multiple regions is a trade-off strategy that tries to deliver the best network performance in terms of network latency and packet loss at the lowest possible cost. Due to the varying costs for data traffic flowing out of data centers, data-intensive enterprises supporting processing use cases that require data ingestion and response retrieval often sponsor real-time, multimedia, and social applications that adopt this strategy. Same-region deployments are implemented to optimize the response time of real-time online gaming or financial services applications. Running microservices in only a single region requires using load balancers as edge appliances and re-architecting business applications and web services to be highly available. The cross-region replication of data for backup and restore scenarios must be controlled to avoid both write conflicts and additional data transfer costs [7].

2.2.2. Challenges of Multi-Region Deployments

The considerations for the deployment and operation of microservices across data center regions are determined by geographic distance, network transmission capacity, and data traffic costs, as well as the cloud data centers where a significant portion of customers are present or distributed. Compute services designed to process and store critical data at the edge of the cloud must be placed with high availability and low response latency. The configuration and usage of scaling groups to manage a scalable deployment of front-end, middle-tier, and back-end services, along with the handling of global name resolutions for DNS requests, require careful consideration of the data center region locations and costs. Certain technologies tend to be deployed closer to the users. By dividing the cloud into four latency tiers ranging from 1.60 ms to 17.88 ms, two-tiered pricing provides greater economic incentives for applications to use compute services operating within commuting distance for most Internet users. The primary technical trades to consider are network performance, expenses, the time to market for global access and traffic management. Responses are pre-fetched and cached.

3. Scalability in Cloud Architectures

One of the most appealing properties of clouds is that they can auto-scale essentially all of the PaaS resources and the virtual machines that host the microservices. This elasticity of the cloud springs from the ability to control all resources using APIs that allow resources to be monitored through cloud management libraries. Due to cloud auto-scaling, it is easy to exploit the behavior of the microservices with the greatest latency. Such latency can be dynamically extracted and used as the input to the traffic manager to redirect requests to microservice instances with the lowest reported latency, thus achieving essentially the same behavior as a web cache, but in a generic way that covers essentially all microservices, independently of the business function that they implement and of which latencies one is measuring in their response time [8]. Service-oriented architectures can be supported using a cloud architecture by hosting separate services in different containers or virtual machines and using a management tool for the automatic configuration of infrastructure for scalability. Such support comes from intuitive properties of clouds that are easy to deploy but would be central and time-consuming to develop starting from network data centers or rented infrastructures based on VMs used in isolation.

3.1. Horizontal vs Vertical Scaling

Nowadays, when we talk about scaling, we usually refer to horizontal scaling. Simply speaking, when we talk about vertical scaling, we mean increasing the capacity of a single machine, for example, a database or an application server, by switching to a more powerful machine. When we refer to a horizontal machine, we mean adding more machines to distribute the load. In some cases, vertical scaling can still make sense. First of all, sometimes reprogramming existing applications or databases to use horizontal scaling can be too expensive. High-performing database clusters are harder to set up, at least in the common case, and potentially introduce other operational risks. Another common argument for vertical scaling is that, at least in theory, a wide range of problems can be solved by using super-powerful mainframes since the use of clusters of less powerful machines will require solving two problems that were not present in the other approach. Of course, technology now allows us to group thousands of computers, while mainframes usually contain only a few powerful nodes, and above all, have the costs of vertical scaling [9].

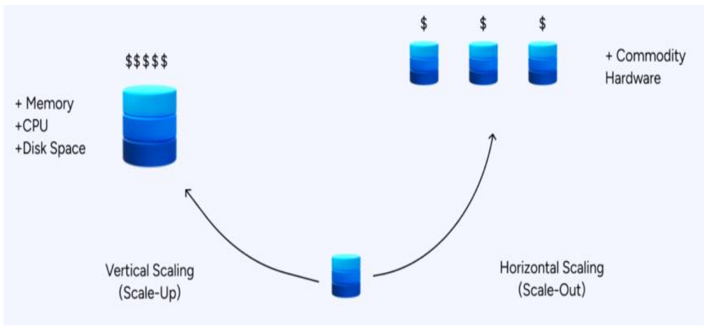

Figure 2. Cloud Scalability: Horizontal vs. Vertical Scaling

3.2. Auto Scaling Techniques

Cloud service providers offer various tools to accomplish auto-scalable systems, given that it is one of the reasons to use their cloud offerings. With auto-scaling, cloud providers keep the specified number of instances for a service running at all times. Three straightforward ways to accomplish auto-scaling are instance metrics or load-based auto-scaling, schedule-based auto-scaling, and one-time auto-scaling.

3.2.1. Load-based or Instance Metrics Auto Scaling

This technique is a way to dynamically increase or decrease the resources allocated to a deployed application. It uses application and service-driven metrics to modify the compute nodes of a cloud-based instance. It has the advantage that over-provisioning can be prevented. This allows the user to save on resources and costs. Applications such as backend databases or caches can benefit from this type of concern about unnecessary data in the cloud. Common solutions include using efficient serialization of subsets of datasets. Remember that hanging data storage will increase both the cost and complexity of your designs. Each time you need to update the cloud schedule, or once the cloud has a new modification, you need to perform extensive tests [10].

3.2.2. Schedule-based Auto Scaling

Schedule-based scaling is a type of scaling that makes use of a preconfigured schedule to start and stop resources. The schedule is useful for managing predictable load increases using cloud resources. This has the advantage that each service can use the resources only when they need them, which can contribute to saving on costs. On the other hand, one aspect that schedule-based scaling does not have is the ability to predict future changes, for example, intensive use of the system in clusters that must be established in the future. Another point is that an unexpectedly high load value may be present, unfortunately making your system unavailable.

4. Security Considerations in Cloud Architectures

To secure their architecture, deployers need to start planning for security from the beginning. Cloud deployment replaces physical security with cloud security, further complicated by the complexity of architectures. Microservices architectures have many interacting but discrete parts, so any compromise, especially a data breach, can potentially affect much of the whole environment. Autoscaling and traffic management mean that components can be short-lived and have a much wider geographical spread, making them more vulnerable to localized attacks. To further protect the environment, the best practice is to protect the infrastructure using a bastion host and security groups. Protect the data and protect the traffic by using a VPN [11].

Guidelines include protecting the environment itself and controlling access for both network access control and logical access, which assigns roles to users. For data, both data at rest and data in transit can be protected through private cloud solutions, privately hosted zones, direct connections, and security policies that assign data control roles to users. Finally, protect the microservices with security groups to safeguard the infrastructure that allows connection on a specific port.

Virtual machines need to be protected through network security groups to limit connections based on origin and port, while data management solutions allow for securing data by restricting access on a data storage account to specific networks. Traffic can then be funneled through a VPN. For user and service access, resource groups and virtual networks are used to control logical access to an interface to a microservice, while data management solutions can be used to allow secured communication between microservices.

4.1. Identity and Access Management

When you create a new account using AWS Organizations or in a standalone account, most of the permissions for that account are managed by using AWS Identity and Access Management (IAM). IAM enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. By default, IAM users do not have the permissions to directly perform actions in AWS. Owners of an account, in this case, the root user, may grant permissions for IAM users to perform specific operations. This is managed by access policies. An access policy is a set of permissions that are attached to an identity and grants permissions to make AWS service requests. Most commonly, these levels of access are managed by using an IAM policy. An IAM policy is a document that contains a collection of permissions that can be assigned to an identity. These permissions describe what actions are allowed or denied on specified resources. In addition to various options for allowing and disallowing actions, explicit policies can also leverage the combination of resources, conditions, and principles [12].

4.2. Data Encryption

Data can be encrypted in three types of states: data addressed to the client, data in transit, and stored data.

Benefit:

- Prevent leaking of important customer information by encrypting data at the state of data storage.
- Effectively protect your data from internal threats and attacks by encrypting data from your services in real-time.
- Simplify the task of administering encrypted data, including key management and processing permissions.

Any managed service or application that needs to query or use the encrypted data can use the client-side master key to create and use keys for these.

By default, when your data is stored physically, it gets encrypted. However, by using the client-side master key creation, this client-side master key creates a simple way to protect your data in the future if you enable "Wallet library" in your managed service settings for this product.

You can enable "Wallet library" by processing the client-side master key to ensure that data encryption service operations are simply added and seamlessly integrated into the architectural components working with your data.

5. Multi-Region Deployments

Resiliency in the cloud is (or at least should be) a core tenet when designing for and deploying your architecture. I can’t say how many times I’ve reviewed someone’s cloud architecture only to find a single region deployment, sometimes without any sort of disaster recovery in place. Availability Zones are great when possible, as they provide both fault tolerance in addition to the resilience of your cloud services. Yes, this can increase costs, but often when we are talking about mission-critical systems, "the costs are too high" is no longer a tenable excuse. One critical discussion that is often left out is the guidance being given to customers on the choice of regions. Vendor-specific guidance often comes way too late [13]. On too many occasions, I have been in conversations where technical and product teams have made commitments about businesses servicing their customers' data residency, availability, and latency requirements in regions that have not yet been vetted or approved by the company's legal and compliance team. Planning for multi-region architecture must be vetted before and not after the cloud architecture has taken on a life of its own. Lastly, don't hide behind the "we'll know where the customer's data lives once the request is made" fallacy. Customers want to know where their data is going to be housed. You should be doing business in countries where transparency is required, but regardless of the legal drivers, doing what is right protects not only your customers but also your interests.

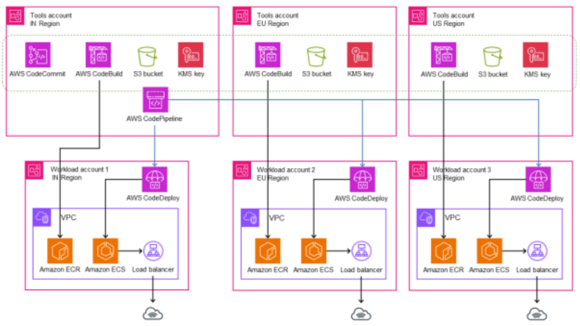

Figure 3. Deployments of Microservices to Multiple Accounts and Regions by Using AWS Code Services and AWS KMS Multi-Region

5.1. Advantages and Challenges

In comparison to the simple single-region application infrastructures, multi-region deployments open up additional advantages and challenges for supporting microservices within the enterprise. On the positive side, the use of multiple regions helps to significantly improve the application's high availability and performance, with sub second response times for failure recovery managed by the AA and grace time-based traffic failback executed by the AMS [14]. An additional boost to availability can be achieved by embedding traffic management intelligence with health metric alarms within sporadically connected networking policy, which will effectively prevent original retries to failed back-to-home from ever reaching the recently failed embedded-to-spore, not stretched alive or alive endpoints.

However, multi-timezone deployments also expose different application infrastructures to additional types of edge failure such as network link, data store, and software library failures, which DNS-only traffic management cannot always gracefully secure or cost-effectively recover from, often reducing the dependency complexity on the client and helping large external load distributed serving the approach they deem known to be preferable for achieving high application availability. Also, replicating stateful memories like database and cache clusters and managing transactions originating from client applications that are bounded by regions require carefully designed distributed software that operates at the same time, which will identify application high availability and recoverability problems not present in stateless microservices and possibly need a different microservices architecture. If more advanced high availability and security response automation is not available or intelligence-enabled with sporadically connected networks and directly embedded networking policy, which must be configured for the MDs to maintain SLAs, there is an additional risk of recent changes made to the application infrastructure, application software, data, or integration endpoints causing the introduction of application availability [15].

Equation 2: Auto Scaling

Scaling Trigger Equation:

$$\text{Scale}_{\text{decision}} = \text{Load}_{\text{current}} - \text{Load}_{\text{threshold}}$$
$$\text{If Scale}_{\text{decision}} > 0,\ \text{scale out; if } < 0, \text{ scale in.}$$

Utilization Metric:

$$\text{Utilization} = \frac{\text{Current Resource Usage}}{\text{Resource Capacity}} \times 100\%$$

Scaling Policy Formula:

$$\text{Scale}_{\text{action}} = \begin{cases} \text{Add Instances} & \text{if Utilization } > \text{ Upper Threshold} \\ \text{Remove Instances} & \text{if Utilization } < \text{ Lower Threshold} \end{cases}$$

5.2. Best Practices in Azure and AWS

To leverage the benefits of cloud computing, especially for secure and cost-effective deployments, good practice in both conditions is required. A plethora of resources regarding best practices in cloud computing and architecture are already available. However, while general perspectives of information technology are useful, such perspectives tend to be abstract, may apply only to traditional infrastructures due to agility constraints, and lack details on how to configure a cloud service—such as a microservice architecture with many building blocks and dependent services, building and deployment services, error handling, a RESTful web API, third-party services, consumable data, etc [16].

In this chapter, we focus on cloud platforms, the most prominent and feature-rich cloud platforms. First, we give a detailed look at the features provided; fees and costs; different on-demand options, including reservation, and bring your license; options to achieve maximum availability (high availability and disaster recovery); and previews and beta programs. Such an overview is a typical starting point for selecting the most appropriate cloud platform in a workload assessment. Second, we work through a reference end-to-end architecture similar to the one shown in the last chapter and identify best practices, recommendations, and items we can easily mix and match—very similar definitions, although with potential differences in naming or details—and names that might refer to alternative or different implementations. Finally, we compare the total costs of running and maintaining a limited or broader selection of services. This can provide a first roadmap for software developers and our solutions personnel to create a service provider shell.

6. Traffic Management in Cloud Architectures

To understand service-specific traffic requirements and related plugins that can be utilized in a microservices network, it is important to consider service-based traffic management with automation. This requires a proper understanding of the microservices-based traffic patterns involving REST and gRPC that can influence service-specific traffic management. The microservices that can be deployed in public clouds are secured and considered to be scalable with backend services, managed using appropriate services. The zone awareness and IP fixed microservices deployments are also explored for Kubernetes clusters. In this chapter, we will review traffic management in cloud architectures. We have considered the microservices and their common traffic patterns that are deployed across multiple regions of our public cloud accounts [17].

The required infrastructure was achieved with the help of cloud-specific managed services for microservices that were based on Kubernetes and cloud-native concepts. These services are unique and targeted to provide a quick troubleshooting experience. The independent deployments cover the services, self-hosted applications, or microservices that provide easier management and configurations in less time. The informative schema covers the deployment model for secure microservices. The lack of shift in microservices that involve front-end, middle-tier, and backend databases is considered a network scalability and traffic issue. The microservices in popular public cloud providers are utilized with the necessary traffic rules. This requires a few missing links when exposing the services outside a cluster with specific DNS settings.

6.1. Load Balancing Techniques

Web servers receive requests from clients and return web pages, often containing the response to those requests. A single web server can only handle a limited number of requests within a given timeframe before it becomes overwhelmed by the number of requests and is unable to respond promptly. Hardware and software mechanisms capable of distributing incoming requests across a pool of servers have been in use since the early days of the World Wide Web. Application layer load balancing is the task of scheduling client requests to a group of servers to improve response time and avoid overloading any one server. The goal is to balance the load both within the group of servers and among other services within the same data center. Perform traffic management to route requests to the closest server. Rewrite the URL to route a client request to a server from another data center. More sophisticated means of balancing load enable receiving servers to take into account both the computational complexity of requests and the overall server load. These mechanisms balance the load more effectively between servers that cache a data source and servers that generate data on demand [18].

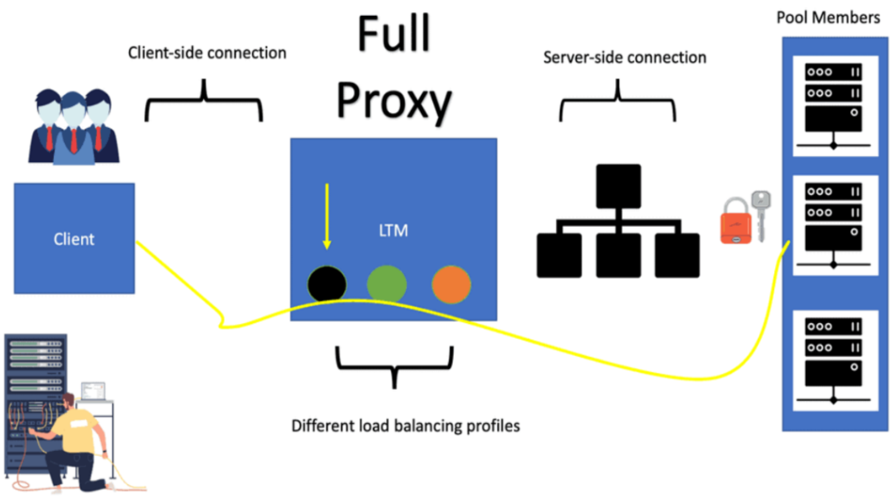

Figure 4. Load Balancing and Scale-Out Architectures

6.2. Content Delivery Networks

Content delivery networks (CDNs) are one of the most critical components in a high-scale and secure cloud-based solution. CDNs help optimize user access to business-critical resources, providing users with a better experience through reduced latency, larger bandwidth, and offloading of workloads to edge servers. The major assets of a modern CDN are the existence of many points that can better prioritize access to edge servers while also providing improved network performance, i.e., more "bang for the buck." Through the use of various cloud services, backed by a cloud provider's points and ubiquitous edge server network, and integrated with hosting services, it is possible to uniformly distribute static resources and dynamic application traffic across a multi-region service deployment. The edge server network that backs CDNs in major cloud providers is sufficiently extensive to make these services capable of effectively executing failover tests based on managed endpoints, prompting an outage in network access or other issues that run intermittently in the service production environment.

Instead of a "clean cutover" in a cutover event, these failover management capabilities may be continuously tested and evaluated as part of a service's continuous deployment lifecycle. In an A/B test scenario, users may be directed to the newly deployed widgets on an experimental edge server or servers while the bulk of user traffic would flow to the "standard widget" distributed across the CDN. Only after various QA and other load and performance testing is done, can the weight of the traffic management be turned to flow more traffic to the newly deployed services. While not super useful if a service is deployed to many endpoints necessary to enable larger regional deployments, it is extremely useful in the earlier stages of development while it is still cost-effective to make use of more compact service deployments [19].

7. Case Studies

We presented architectural patterns for various cross-cutting concerns. In this section, we assess the scalability and business continuity capabilities of our patterns through two case study architectures. Next, we discuss the results of our performance evaluation.

The case studies are drawn from our practice experience with application deployments for enterprises in various business verticals. We have created architectures for end customers in three domains: e-commerce, sales and marketing, and security monitoring. These domain solutions use services from a variety of cloud vendors to provide a wide diversity of business domain functionalities and non-functional qualities, appealing to customers of vastly varying sizes.

Each case study is presented with attention to its business, development, and business continuity context. The case studies are selected to show not only the scaling of workloads across a cloud vendor's data centers but also the effects of scale, such as the development and operational effort involved. Once the case studies are presented, we provide a comparison and lessons learned from the rapid iterations that clients have had to make to correct missing parts [20].

7.1. Real-world Implementations in Azure and AWS

In this section, I provide a selection of real-world deployment strategies for cloud architectures built on Azure and AWS. These leading cloud providers offer a wide range of network services, repackaging many traditional infrastructure devices including dedicated firewalls, VPN connections, load balancers, and more. Often, these cloud features allow one to construct powerful high-availability virtual network deployments with only minimal experience in routing concepts. If a larger install base is desired, especially if there are no such experiences with cloud networks, the runtime configuration on a CDN seems to be a better solution anyway. We look at zero-hop deployments to multi-region load-balanced back ends based on geo-optimized DNS.

Our last real deployment example serves as an astronomy pipeline. It handles a great number of short requests that do not fit well with the caching nature of a CDN. This fact made it very attractive for a simple two-instance deployment over Azure and a CDN. Because of its open API, the provisioning and configuration of additional instances were automated. Nowadays, this task could also be performed from an Auto-Scale service without the need to access the CDN separately. Refactorings with caching and App

Service and its built-in scale capability allow us additionally to employ more flexible scaling strategies.

8. Conclusion

Concerns in designing both scalable-point online traffic management and regional distributed cloud architecture were discussed. Microservice programming, autoscaling, building pipeline, storage transfer, build, upload, and route operation phases of the microservices, and the security and performance aspects were discussed in detail. The routines and working style of the packet load balancer, and elastic cache services, as well as the processing mechanisms of Identity and Management systems, were explained. As a third design cycle, the technology-stack process, which went through the front-end validations of platforms, was shared [21].

Getting three paths in one product and a scalable architecture allocation in cloud service fabric are offered. Cloud services are provided with both multi-region deployment interaction management made up of an existing deployment functionality and for each function running on that is a part of a flexible page workload grade, scale-out by making it work very smartly in the presence of it with minimum and the maximum number and warming policies, alternative and draw pipelines, storage transfer, and fun build, upload, and routing operations. It is ready to be used in the world's secure and high-level programs and to cover the repository list for deployment with the packet load generator and elastic cache services that help the functions, identity, container storage, and management systems that ensure privacy and peace.

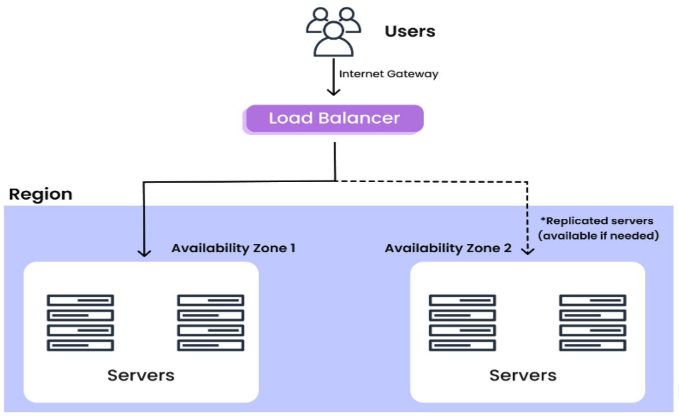

Figure 5. AWS vs Azure: High Availability

8.1. Summary of Key Findings

The deployment of the designed system on both the Azure and AWS platforms has shown that the specified approach is working and results in a highly scalable, secure, and reliable cloud architecture that allows providing scalable access to the computing resources of PaaS and a fault-tolerant operation of the whole system. The comparison of the automated deployment to several regions in both Azure and AWS with a manual modification of the settings presents the results of the comparison of the performance of both cloud systems for the different architectures of an auto-scaling group that is used in rescaling the count of the microservices' instances when a high load is detected.

During the deployment of the designed microservices on both Azure and AWS, three regions in Europe, two regions in North America, and two regions in Asia were used. The two regions in each of the three large regions allow deploying the latency-based routing system that is aware of the presence of at least four main facilities in Europe, two facilities in North America, and two facilities in Asia. If the deployed services have been divided into several layers, the latency-based routing system is an economical and technically simple solution that allows the distribution of the overall load between all possible facilities [22].

Equation 3: Traffic Management

Load Distribution Equation:

$$\text{Load}_{\text{balance}} = \frac{\text{Total Incoming Requests}}{\text{Number of Instances}}$$

Routing Decision Formula:

$$\text{Route}_{\text{decision}} = \text{Traffic}_{\text{incoming}} \times \text{Routing}_{\text{policy}}$$

Weighted Routing Strategy: For multiple service versions:

$$\text{Traffic}_{\text{version}} = \text{Total Traffic} \times \text{Weight}_{\text{version}}$$

8.2. Future Research Directions

In this chapter, we have summarized some of the past and ongoing research in deploying scalable and secure microservice architectures on public cloud providers. This research cuts across multiple subfields within cloud architecture and infrastructure, with design and deployment challenges emerging in specific areas such as identity and resource management for microservice-based applications or addressing these challenges as part of implementing complete layered multi-region deployments of microservices hosted on service and container instantiations combined with custom virtual private networks. In future work, the research and practitioner communities should consider addressing individual architectural layers directly at the hyper-scale. The general stages of our techniques are deployable in the commercial hyperscale settings of public clouds. However, the scale of investment and focus required to leverage these techniques successfully requires sizable payouts from applications that process hundreds of millions of monthly and peak users, rather than dozens. This workload scale reduces the availability of affordable test subjects for general cluster or service virtualization information flow research.

References

[1] Syed, S. (2022). Breaking Barriers: Leveraging Natural Language Processing In Self-Service Bi For Non-Technical Users. Available at SSRN 5032632.

[2] Nampally, R. C. R. (2022). Neural Networks for Enhancing Rail Safety and Security: Real-Time Monitoring and Incident Prediction. In Journal of Artificial Intelligence and Big Data (Vol. 2, Issue 1, pp. 49–63). Science Publications (SCIPUB). https://doi.org/10.31586/jaibd.2022.1155

[3] Danda, R. R. (2022). Innovations in Agricultural Machinery: Assessing the Impact of Advanced Technologies on Farm Efficiency. In Journal of Artificial Intelligence and Big Data (Vol. 2, Issue 1, pp. 64–83). Science Publications (SCIPUB). https://doi.org/10.31586/jaibd.2022.1156

[4] Rajesh Kumar Malviya , Shakir Syed , RamaChandra Rao Nampally , Valiki Dileep. (2022). Genetic Algorithm-Driven Optimization Of Neural Network Architectures For Task-Specific AI Applications. Migration Letters, 19(6), 1091–1102. Retrieved from https://migrationletters.com/index.php/ml/article/view/11417

[5] Patra, G. K., Rajaram, S. K., Boddapati, V. N., Kuraku, C., & Gollangi, H. K. (2022). Advancing Digital Payment Systems: Combining AI, Big Data, and Biometric Authentication for Enhanced Security. International Journal of Engineering and Computer Science, 11(08), 25618–25631. https://doi.org/10.18535/ijecs/v11i08.4698

[6] Syed, S. (2022). Integrating Predictive Analytics Into Manufacturing Finance: A Case Study On Cost Control And Zero-Carbon Goals In Automotive Production. Migration Letters, 19(6), 1078-1090.

[7] Nampally, R. C. R. (2022). Machine Learning Applications in Fleet Electrification: Optimizing Vehicle Maintenance and Energy Consumption. In Educational Administration: Theory and Practice. Green Publication. https://doi.org/10.53555/kuey.v28i4.8258

[8] Danda, R. R. (2022). Application of Neural Networks in Optimizing Health Outcomes in Medicare Advantage and Supplement Plans. Journal of Artificial Intelligence and Big Data, 2(1), 97–111. Retrieved from https://www.scipublications.com/journal/index.php/jaibd/article/view/1178

[9] Chintale, P., Korada, L., Ranjan, P., & Malviya, R. K. (2019). Adopting Infrastructure as Code (IaC) for Efficient Financial Cloud Management. ISSN: 2096-3246, 51(04).
[10] Kumar Rajaram, S.. AI-Driven Threat Detection: Leveraging Big Data For Advanced Cybersecurity Compliance. In Educational Administration: Theory and Practice (pp. 285–296). Green Publication. https://doi.org/10.53555/kuey.v28i4.7529
[11] Syed, S. (2022). Leveraging Predictive Analytics for Zero-Carbon Emission Vehicles: Manufacturing Practices and Challenges. Journal of Scientific and Engineering Research, 9(10), 97-110.
[12] RamaChandra Rao Nampally. (2022). Deep Learning-Based Predictive Models For Rail Signaling And Control Systems: Improving Operational Efficiency And Safety. Migration Letters, 19(6), 1065–1077. Retrieved from https://migrationletters.com/index.php/ml/article/view/11335
[13] Danda, R. R. (2022). Deep Learning Approaches For Cost-Benefit Analysis Of Vision And Dental Coverage In Comprehensive Health Plans. Migration Letters, 19(6), 1103-1118.
[14] Sarisa, M., Boddapati, V. N., Kumar Patra, G., Kuraku, C., & Konkimalla, S. (2022). Deep Learning Approaches To Image Classification: Exploring The Future Of Visual Data Analysis. In Educational Administration: Theory and Practice. Green Publication. https://doi.org/10.53555/kuey.v28i4.7863
[15] Syed, S. (2022). Towards Autonomous Analytics: The Evolution of Self-Service BI Platforms with Machine Learning Integration. Journal of Artificial Intelligence and Big Data, 2(1), 84-96.
[16] Nampally, R. C. R. (2021). Leveraging AI in Urban Traffic Management: Addressing Congestion and Traffic Flow with Intelligent Systems. In Journal of Artificial Intelligence and Big Data (Vol. 1, Issue 1, pp. 86–99). Science Publications (SCIPUB). https://doi.org/10.31586/jaibd.2021.1151
[17] Ramanakar Reddy Danda. (2022). Telehealth In Medicare Plans: Leveraging AI For Improved Accessibility And Senior Care Quality.
[18] Venkata Nagesh Boddapati, Manikanth Sarisa, Mohit Surender Reddy, Janardhana Rao Sunkara, Shravan Kumar Rajaram, Sanjay Ramdas Bauskar, Kiran Polimetla. Data migration in the cloud database: A review of vendor solutions and challenges. Int J Comput Artif Intell 2022;3(2):96-101. DOI: 10.33545/27076571.2022.v3.i2a.110
[19] Syed, S. (2021). Financial Implications of Predictive Analytics in Vehicle Manufacturing: Insights for Budget Optimization and Resource Allocation. Journal Of Artificial Intelligence And Big Data, 1(1), 111-125.
[20] Syed, S., & Nampally, R. C. R. (2021). Empowering Users: The Role Of AI In Enhancing Self-Service BI For Data-Driven Decision Making. In Educational Administration: Theory and Practice. Green Publication. https://doi.org/10.53555/kuey.v27i4.8105
[21] Danda, R. R. (2021). Sustainability in Construction: Exploring the Development of Eco-Friendly Equipment. In Journal of Artificial Intelligence and Big Data (Vol. 1, Issue 1, pp. 100–110). Science Publications (SCIPUB). https://doi.org/10.31586/jaibd.2021.1153
[22] Chandrakanth Rao Madhavaram, Eswar Prasad Galla, Hemanth Kumar Gollangi, Gagan Kumar Patra, Chandrababu Kuraku, Siddharth Konkimalla, Kiran Polimetla. An analysis of chest x-ray image classification and identification during COVID-19 based on deep learning models. Int J Comput Artif Intell 2022;3(2):86-95. DOI: 10.33545/27076571.2022.v3.i2a.109