Shodhshauryam, International Scientific Refereed Research Journal
Available online at : www.shisrrj.com
OPEN ACCESS
© 2024 SHISRRJ | Volume 7 | Issue 2
ISSN : 2581-6306
doi : https://doi.org/10.32628/SHISRRJ

Scheduling Web-Jobs with Azure App Service

K. Naresh[1], P. Charitha[2]

[1]Assistant Professor, Department of MCA, Annamacharya Institute of Technology & Sciences, Tirupati, Andhra Pradesh, India

[2]Post Graduate, Department of MCA, Annamacharya Institute of Technology & Sciences, Tirupati, Andhra Pradesh, India

Article Info

Publication Issue :
March-April-2024
Volume 7, Issue 2

Page Number : 351-356

Article History
Received : 15 March 2024
Published : 30 March 2024

ABSTRACT

The cloud is transforming application security and design. Applications are separated into more manageable, dispersed services as opposed to existing as monoliths. These offerings can interact with one another via APIs, eventing, or asynchronous messaging. Applications create new occurrences in a horizontal manner in response to demand.

Web apps, REST APIs, and mobile back ends may all be hosted on Azure App Service, an HTTP-based platform. The language of your choosing is yours to advance in. Python, PHP, Node.js, Java, and.NET Core are a few examples. Applications run smoothly and scale both on Windows and Linux-based computers.

Furthermore, you can benefit from its DevOps features, which include Azure DevOps, GitHub, Docker Hub, and other sources for package management, staging environments, custom domains, TLS/SSL certificates, and continuous deployment. The cost of the Azure computing resources you utilize when you Apply App Service. The App Service package you choose to use your Apps decide how much computational power you utilize.

Objective: There are different application states. Both synchronous and asynchronous operations are carried out. Apps need to be able to bounce back quickly from errors. Attackers target applications all the time. Automated and predictable deployments are required. To understand the system, telemetry and monitoring are essential.

Keywords : Cloud computing, Application architecture, Microservices, Azure App Service, Serverless computing, DevOps, Continuous deployment, Resilient applications, Security, Monitoring and telemetry, Functions, Logic Apps, Web Jobs, Literature survey.

I. INTRODUCTION

This project aims to explore and implement the scheduling of WebJobs within Azure App Service. By leveraging the capabilities of Azure App Service and WebJobs, organizations can automate routine tasks, process large volumes of data, and perform various backend operations without the need for dedicated infrastructure or complex setups. Scheduling WebJobs enables businesses to execute critical tasks at predefined intervals, ensuring timely data processing, maintenance activities, and system updates.

The objective of this project is to demonstrate how to effectively schedule and manage WebJobs within Azure App Service, covering aspects such as task automation, job scheduling, error handling, and monitoring. By implementing a well-defined scheduling mechanism, organizations can enhance operational efficiency, reduce manual intervention, and streamline their workflow processes.

II. LITERATURE REVIEW

Examining the Literature on Scheduling Web-Jobs with Azure AppService

several key themes and findings emerge from existing research. Scholars have explored the various methods and best practices for scheduling tasks within the Azure AppService environment, highlighting the importance of efficient resource utilization, scalability, and reliability. Furthermore, researchers have investigated the role of DevOps practices in managing and scheduling WebJobs, emphasizing the need for continuous deployment, monitoring, and performance optimization.

An Overview of the technology

Microsoft Azure offers (PaaS) that makes it simple for customers to launch and maintain mobile back ends, REST APIs, and web apps. Azure App Service frees developers from the burden of managing infrastructure so they can concentrate on creating applications.

For the project of scheduling web jobs with Azure App Service, the following steps can be outlined:

Establishment of Azure App Service: To begin, open the Azure portal, Set up an instance of Azure App Service ,and select the appropriate configuration based on the specifications of your project.

Development Web Jobs: Develop web jobs that need to be scheduled. These can be scripts or code snippets that perform specific tasks or run at scheduled intervals. Ensure that the web jobs are compatible with the chosen App Service environment.

Deployment of Web Jobs: Once the web jobs are developed, deploy them to the Azure App Service instance. This can be done using various deployment methods supported by Azure, such as Git deployment, Azure DevOps pipelines, or directly from Visual Studio.

Configuration of Job Scheduler: Azure App Service provides a built-in job scheduler that allows you to schedule the execution of web jobs at specified intervals or times. Configure the job scheduler to run the deployed web jobs according to the desired schedule.

Monitoring and Management: Monitor the execution of scheduled web jobs using the logging and monitoring features provided by Azure App Service. This allows you to track the

execution status, identify any errors or issues, and troubleshoot as needed.

Scaling and Optimization: As the project evolves and the workload changes, consider scaling and optimizing the Azure App Service instance to meet the growing demands. This can involve adjusting the pricing tier, scaling up or out based on resource requirements, and optimizing performance.

Scheduling Web-Jobs with Azure AppService

One of the key advantages of utilizing Azure App Service is its integration with the broader Microsoft Azure ecosystem, offering features like load balancing, automated administration, security, and autoscaling. By leveraging these features, the project ensures the reliability, scalability, and security of the scheduled web jobs. Moreover, Azure App Service facilitates seamless integration with DevOps practices, enabling continuous deployment using GitHub, Docker Hub, Azure DevOps, and more resources. This ensures that the deployment process is automated, predictable, and efficient.

Additionally, Azure App Service provides essential tools for managing application states, handling operations in parallel and asynchronously, and ensuring resilience in the face of failures. This aligns with the project's objective of building resilient, distributed applications that can withstand various challenges and threats. Furthermore, Azure App Service offers comprehensive observing and telemetry competencies, enabling developers to gain valuable perceptions of the system's performance as well as behavior.

III. METHODOLOGY

Approach

Azure App Service offers various features that facilitate scheduling and running web-jobs efficiently. Additionally, Azure Application Service integrates seamlessly with DevOps practices, allowing for continuous deployment from sources such as Azure DevOps, GitHub, and Docker Hub.

Involves utilizing Azure Functions and Logic Apps, which are serverless services provided by Azure, to streamline and automate tasks within the web-jobs. Azure Functions enable serverless compute, allowing developers to execute code in response to events without managing infrastructure. Conversely, a serverless process integration platform is offered by Azure Logic Apps for orchestrating workflows and integrating various services.

Implementation

Security, load balancing, autoscaling, and automated administration are just a few of the many capabilities that Azure App Service offers to improve the functionality and efficiency of hosted apps.

Developers may use the WebJobs feature in the Azure AppService project to schedule web-jobs. This feature enables the execution of scripts or code inside the App Service web application. The framework created especially for WebJobs, the WebJobs SDK, makes it easier to write the code needed to react to events in Azure services.

Characteristics

Scalability: Azure App Service allows for easy scaling of web applications and web jobs, ensuring that they can handle varying workloads efficiently.

Flexibility: With Azure App Service, developers have the flexibility to choose their preferred programming language for developing web jobs.

Serverless Architecture: Leveraging serverless computing with Azure Functions or Logic Apps allows for efficient execution of tasks without the need to provision or manage servers, reducing operational overhead.

Integration: Azure App Service integrates seamlessly with other Azure services, allowing developers to incorporate additional functionalities such as storage, databases, messaging, and more into their web jobs as needed.

Automated Deployment: Developers can implement automated deployment pipelines, ensuring consistent and predictable deployment of web jobs.

Security: To guard against unwanted access and security breaches, Azure App Service has integrated security measures, such as role-based access control, TLS/SSL certificates, and network isolation for web tasks and data.

Cost-effectiveness: With Azure App Service, developers only pay for the compute resources they usage, which makes it an affordable option for hosting and scheduling web jobs without the need to manage underlying infrastructure.

Community Support: Azure App Service advantages from a strong and vibrant development community and resources, providing support, tutorials, and best practices for building and scheduling web jobs effectively.

Data Preprocessing

In the context of scheduling web-jobs with Azure AppService, data preprocessing involves preparing and cleaning the data that will be used by the web-jobs. This could include tasks such as removing duplicates, handling missing values, transforming data into the required format, and ensuring data quality before it is used for scheduling tasks within the Azure AppService environment.

Preprocessing Data

Preprocessing Data for Scheduling Web-Jobs with Azure AppService project involves preparing the data needed to schedule and run web jobs effectively within the Azure cloud environment.

In addition to offering flexibility in development, improves reliability and scalability. Constant implementation from sources like GitHub, Docker Hub, and Azure DevOps is made possible by Azure App Service's seamless integration with DevOps tools. Deployment procedures are further simplified by its support for TLS/SSL certificates, custom domain administration, and setting up scenes.

IV. EXPERIMENTAL SETUP

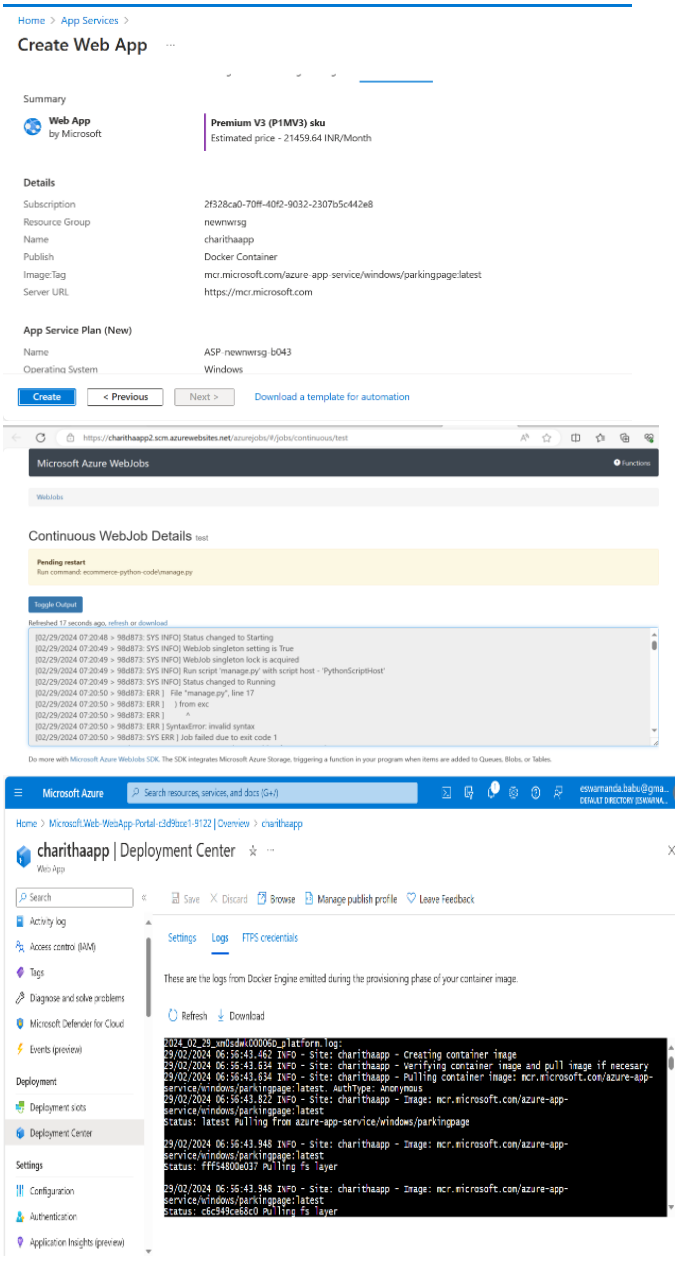
An entirely managed HTTP-based solution is called Azure App solution offered by Microsoft Azure for hosting a wide range of applications, including mobile back ends, REST APIs, and web apps. Together support for multiple programming languages and environments, including Windows and Linux,Azure App Service lets developers create, implement, and expand applications with ease. With a Users only pay for the computational resources they utilize under the pay-as-you-go concept., making Azure App Service a cost-effective solution for hosting applications in the cloud.

WebJobs Integration: Within the Azure App Service project, WebJobs will be integrated to facilitate the execution of background tasks. WebJobs enable the running of scripts or code within an App Service web application, providing a mechanism for performing various operations asynchronously.

The WebJobs SDK will be utilized to simplify the development of background tasks within the Azure App Service project. This framework streamlines the process of responding to events in Azure services, enhancing the efficiency of task execution.

Azure Cloud Service: The experimental setup will leverage Azure cloud services to provide the infrastructure and resources required for hosting the Azure App Service project. Azure's robust cloud platform ensures scalability, reliability, and security for the experiment.

V. ANALYSIS



VI. DICUSSIONS

Scheduling Web-Jobs with Azure appservice Implications

With Azure DevOps integration, continuous deployment pipelines streamline development processes, allowing for easy updates and enhancements. Additionally, Azure App Service enables the execution of background tasks using WebJobs, which can be seamlessly integrated into web applications to perform scheduled or on-demand operations. This integration enhances application functionality and efficiency while reducing manual intervention.

Benefits and drawbacks

Benefits

1. Cloud App Deployment Made Easier

It only takes a few clicks to deploy your application to Azure if you are using Visual Studio. Moreover, you may use Powershell, Git, and other tools to automatically deploy your application. Azure takes care of high availability and multi-server code deployment automatically.

2. Combine Several Apps to Reduce Costs

The ability to integrate numerous programs is one of the nicest features. Prior to the release of Azure Worker Roles, every application needed to have its own server. By merging your application and moving to App Services, you may save a significant amount of money. You may simply place them on distinct App Service Plans, which are essentially different server groups, if you wish to keep them apart.

3. High Availability & Auto-Scaling on Demand

To employ more servers, you may autoscale your application either automatically or manually. The server size and autoscaling policies are set by you based on your App Service Plan. Along with offering a 99.95% SLA, Azure also automatically maintains high availability.

Drawbacks

Performance: Physical servers can frequently outperform virtual servers due to their specialized hardware resources. The application performance of your customers might be optimized by the absence of rivalry or disruption from other servers.

Maintenance: To reduce possible security threats, physical servers need to undergo routine maintenance. Troubleshooting is made easier because upkeep is limited to a single entity.

Dependability: On physical servers, resource availability and allocation are predictable, which leads to improved consistency and dependability. For customers running applications that need for dependably dependable performance, this is the ideal option.

Control: tangible servers provide you and your client total control because they are tangible things. This enables you to customize certain setups or modifications to your clients' requirements.

VII. CONCLUSION

Cloud resource management is a key component of cloud computing service models. Virtual computers, network equipment, load balancers, and firewalls are among the resources that cloud service providers supply to their customers. One of the most important problems with Infrastructure as a Service (IaaS) is resource management. The safety of tangible hardware and appliances at cloud service providers' locations is strongly related to their availability. Attackers get entry to resources that are guarded by taking advantage of faults, configuration errors, system leaks, and design flaws. Tenant computers and compromised virtual machines are vulnerable to denial-of-service attacks. Before utilizing cloud services, cloud adopters must take data security and privacy into account during the adoption and execution of cloud computing. blockchain-based technologies

VIII. REFERENCES

[1] P. T. Jaeger, J. Lin, and J. M. Grimes, "Cloud computing and information policy: Computing in a policy cloud?," Journal of Information Technology Politics, vol. 5, no. 3, pp. 269-283, 2008.

[2] C. Vidal and K.-K. R. Choo, "Situational Crime Prevention and the Mitigation of Cloud Computing Threats," in International Conference on Security and Privacy in Communication Systems, 2017: Springer, pp. 218- 233.

[3] N. Khan and A. Al-Yasiri, "Cloud security threats and techniques to strengthen cloud computing adoption framework," in Cyber Security and Threats: Concepts, Methodologies, Tools, and Applications: IGI Global, 2018, pp. 268-285.

[4] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi, "A risk mitiga- tion approach for autonomous cloud intrusion response system," Comput- ing, vol. 98, no. 11, pp. 1111-1135, 2016.

[5] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network sys- tems," IEEE transactions on dependable and secure computing, vol. 10, no. 4, pp. 198-211, 2013.