PRACTICAL – 9

AIM: **Prepare the detailed case study on Microsoft Azure/AWS DevOps tool in the area of cloud business application.**

CASE STUDY ON AWS:

Introduction:

Service-Oriented Architecture (SOA) may be a kind of software design where services are provided to the opposite components by application components, through a communication protocol over a network. Its principles are independent of vendors and other technologies.

Implementations of SOA vary in terms of granularity: from only a few services that cover large areas of functionality to several dozens or many small applications in what's termed "microservice" architecture.

AWS Lambda:

AWS Lambda may be a service offered by the Amazon Web Services platform. AWS Lambda allows you to upload code that may be run on an on-demand container managed by Amazon. AWS Lambda will manage the provisioning and managing of servers to run the code, so all that's needed from the user may be a packaged set of code to run and some configuration options to define the context during which the server runs. These managed applications are mentioned as Lambda functions.

Modes of Operation:

AWS Lambda has 2 modes of operations.

- Asynchronous/Event-driven
    - Lambda functions is run in response to an occasion in asynchronous mode. Any source of events, such as S3, SNS, etc. won't block and Lambda functions can make the most of this in some ways, like establishing a processing pipeline for a few chains of events.
    - There are many sources of data, and counting on the source events are pushed to a Lambda function from the event source, or polled for events by AWS Lambda.
- Synchronous/Request->Response
    - For applications that need a response to be returned synchronously, Lambda are often run-in synchronous mode.

- Typically, this is often utilized in conjunction with a service called API Gateway to return HTTP responses from AWS Lambda to an end-user, however Lambda functions may be called synchronously via a right away call to AWS Lambda.

AWS Lambda functions are uploaded as a zipper file containing handler code additionally to any dependencies required for the operation of the handler.

Lambda Functions as an Evolution of SOA:

Basic SOA could be a thanks to structure your code-base into small applications so as to learn an application within the ways described earlier during this article. Arising from this, the tactic of communication between these applications comes into focus. Event-driven SOA (aka SOA 2.0) allows for not only the normal direct service-to-service communication of SOA 1.0, but also for events to be propagated throughout the architecture so as to speak change.

Event-driven architecture may be a pattern that naturally promotes loose coupling and composability. By creating and reacting to events, services are often added ad-hoc to feature new functionality to an existing event, and several other events is composed to supply richer functionality.

CASE STUDY ON MICROSOFT AZURE

Introduction:

Azure is a new cloud computing platform under development by Microsoft (microsoft.com/windowsazure). Cloud computing allows developers to host applications in an Internet-accessible virtual environment. The environment transparently provides the hardware, software, network and storage needed by the application.

As with other cloud environments, Azure provides a hosted environment for applications. The added benefit of Azure is that .NET Framework applications can be deployed with minimal changes from their desktop siblings.

Applying service-oriented architecture (SOA) patterns and utilizing the experiences collected when implementing service-oriented solutions will be key to success when moving your services and applications into the new arena of cloud computing. To better understand how SOA patterns can be applied to Azure deployments, let's take a look at a scenario in which a fictional bank moves its services to the cloud.

Performance and Flexibility:

After some stress testing, the Woodgrove Bank development team found that having only one central data store in SQL Azure led to slower and slower response times when traffic increased. The developers decided to address this performance issue by using Azure table storage, which is designed to improve scalability by distributing the partitions across many storage nodes. Azure table storage also provides fast data access because the system monitors usage of the partitions and automatically load-balances them. However, because Azure table storage isn't a relational data store, the team had to design some new data storage structures and pick a combination of partition and row keys that would provide good response times.

Messaging and Queuing:

The objective is that no message should be lost even if services are offline due to error conditions or planned maintenance. The Asynchronous Queuing pattern allows this, though some offerings are not suitable for this pattern. For example, prompt answers with confirmation or denial of money transfers are necessary when dealing with online card transactions. But in another situation the pattern would do fine.

Communication between the Web and Worker roles is done with Azure Queues (as of the November CTP version it is possible to communicate directly between role instances), which are by default both asynchronous and reliable.

Putting Queues to Work:

As soon as a customer sends a message to UserAccountService, this message is placed in a Azure Queue and the customer receives a confirmation message. UserAccountWorker will then be able to get the message from the queue. Should UserAccountWorker be down, the message will not be lost as it is stored securely in the queue

If the processing inside UserAccountWorker goes wrong, the message will not be removed from the queue. To ensure this, the call to the DeleteMessage method of the queue is made only after the work has been completed. If UserAccountWorker didn't finish processing the message before the timeout elapsed (the timeout is hardcoded to 20 seconds), the message will again be made visible on the queue so that another instance of UserAccountWorker can attempt to process it.

As soon as a customer sends a message to UserAccountService, this message is placed in a queue and the customer receives a confirmation message of type TransactionResponse. From the perspective of the customer, Asynchronous Queuing is used. ReliableMessaging is used to communicate between UserAccountStorageAction and AccountStorageWorker, which reside in the Web role and Worker role, respectively.

Processing Messages:

The ProcessMessage method first needs to get the content of the message. This can be done in one of two ways.

First, the message could be stored as a string in the queue.

Second, the message could be serialized XML.

Idempotent Capability:

What if one of Woodgrove Bank's customers sends a request to transfer money from one account to another and the message gets lost? If the customer resends the message, it is possible that two or more of the requests reach the services and gets treated separately.

One of the Woodgrove Bank team members immediately identified this scenario as one that requires the Idempotent Capability pattern. This pattern demands that capabilities or operations are implemented in such a way that they are safe to repeat. In short, the solution that Woodgrove Bank wants to implement requires well-behaved clients that attach a unique ID to each request and promise that they will resend the exact same message including the same unique ID in case of a retry. To be able to handle this, the unique ID is saved in the Azure table storage. Before processing any requests, it is necessary to check if a message with that ID was already processed. If it has been processed, a correct reply will be created, but the processing associated with the new request will not take place.

Although this means bothering the central data store with extra queries, it was deemed necessary. It will result in some deterioration of performance since some queries are made to the central data store before any other processing can take place. However, allowing this to consume extra time and other resources is a reasonable choice in order to meet Woodgrove Bank's requirements.

CONCLUSION:

In this practical, we learned about service-oriented architecture (SOA). We studied about its advantages and disadvantages. We learned about different platforms like AWS offered by Amazon, Azure offered by Microsoft.

Remarks:

Signature:

Marks: