INTRODUCTION

This case is based on Lambdas and how our product MTD (Digital Transactional Engine) can develop and analyze through Lambdas in addition to other AWS services. MTD is a flexible, secure and transactional workflow that supports high volumes of complex operations (high transactionality) that involve the orchestration of multiple systems.

As an additional definition, MTD allow:

- It allows to execute long-running flows (feasible to integrate with human tasks, through APIs) and short-running typically used for orchestration of complex processes.
- It provides an information model that facilitates the traceability and audit of historical processes, as well as online monitoring to have complete control of its processes.
- To facilitate the definition of the processes, it allows to implement the flows based on definitions made in Bizagui Modeler (Free Workflow Construction IDE).
- This solution can be acquired as a server license or in SaaS mode, allowing elasticity and rapid deployment of the processes.

During the development of this SDP, will be able to know the operation and the different components that the MTD possesses as such and how it is related to the different AWS instances and services.

In respect the problems that our solution comes to solve are relates to the following:

- Online control of orders in process.
- Manage orders with exception.
- Process all orders.
- Comply with dispatch times.
- Early warning.
- Timely information.
- Purchase order statistics.
- Capacity against high demand.
- Avoid crafts.
- Reduce operating costs.

For the business, MTD solves the life cycle process of a purchase order, handling communication and integration between the different legacy systems existing in Customer (Gift card, EOM, Paperless, Sales Support) and customer-side interactions, such as the generation of purchase tickets / invoices and the subsequent sending by email, and their operation and traceability is available to Customer executives through a BackOffice Web portal.

Being a cloud-mounted solution, MTD is a highly scalable platform which provides a high level of availability and adaptation to the operating requirements that Customer requires. Our proposal is based on a solution that allows to collect, manage and orchestrate purchase orders or transactions incorporating business rules, monitoring elements and others. An example of its implementation is in the management of Retail Purchase Orders where it manages its flow considering business rules, state management, exception control, mailings and integration with the different legacies among other things. This solution is our Digital Transactional Engine (MTD), mounted on an AWS environment.

For this solution, the services used by AWS Cloud are ***Lambdas*** (among others).

Its logical operation is developed as follows:

When entering a purchase order, it is stored in an S3 bucket in xml format. This activates a lambda that transforms the order to json format and sends the command to a SQS queue and this SQS sends the command to a queue reader. At this point there is a condition that, if the queue reader does not read the order in 10 attempts, it goes to an SQS where the order will be pending. In case the reader correctly reads the order, it will go to an API connected to the database engine.

When the order arrives at the database, it is verified in what state the order is and depending on this, it will be redirected to the corresponding API with the status of the order, launching a lambda and saving the order in the database.

In the event that an order fails when trying to enter the state of broadcast_dte, it will be sent to a step function that will trigger two lambdas where the error will be controlled.

To control errors that occur in the remaining stages, these will be directed to an API that controls these errors by triggering the corresponding lambda by storing and updating the order in the database.

For the second stage of handling the order status, the SES mail service is used, which, at the time of the order, passes through the issuance_dte successfully and is updated in the database, sends a legacy through of an email to the customer indicating that the order began to be processed.

The entire workflow is controlled with CloudWatch metrics and alarms for the different services indicated in the scheme.

The environment is deployed in a mixed On-Premise and AWS architecture and deployment using CloudFormation where the following services are used:

VPC, Subnets, Nat Gateway, Internet Gateway, Transit Gateway, API Gateway, Lambdas, SQS, SES, SNS, ECS, ECR, ALB, EC2, RDS, S3.

As a aws lambda validation AWS lambda is used for processing orders, for example, in the first step, a file is taken from an S3 bucket in format XML, which, is converted to json format and is sent to the SQS queue.

With lambda, the customer is given the processing of purchase orders stored in Amazon S3 buckets from when an order is entered until the order ends its flow, being stored in the PostgreSQL database.

Around 25 lambdas functions are used for the present solution, of which the most relevant are added to the diagram. All these are worked in the us-west-2 region of Oregon.

For each lambda function, executions per month may vary. Depending on the function, some reach 30,000 monthly executions approx., as there are others that can reach 190,000 approx., depending on the type of task you complete. All these executions are controlled with CloudWatch with control metrics and alarms, in addition to having constancy in CloudWatch logs.

The challenge that the Customers proposed to develop the solution was to improve the flow times of a purchase order, which was satisfactorily covered with AWS lambda and additional services, thus increasing the quality and improvements of the system development so that the calls the lambdas will take no more than 5 milliseconds. With this form of work, in which Amazon takes care of the servers, it allowed developers to improve the infrastructure and work of the application.

The deployment of the Lambdas for the presented scheme the "all at once" implementation pattern is used with the eight lambdas functions presented so that all the work is updated and passes to the new version at the same time, without interrupting the flow of the orders.

For the solution, the SAM (Serverless Application Model) implementation application model was used as an architecture framework due to its adaptability with other AWS services such as API, databases, etc., and therefore focus on the application and its development

Every lambda function connects to AWS services, including APIs, SQS services, database, SES mail service and AWS step functions, in addition to being all controlled with CloudWatch.

For the main processing and data flow of the orders within the implemented scheme, the lambdas functions are used due to their adaptation with the form of "Serverless" work without, with this, the developers and support focus on the development of the application and comply the requirements that the Customers requests. In addition, you can better manage the databases used in the solution, due to the feasibility of AWS managing the servers.

The system was designed based on the requirements requested by our Customers and based on the initial architecture provided by the architecture team, achieving a satisfactory work architecture, both for the Customers and for the development and support team.

In respect the service requirement / solution characteristics**,** for the stress tests carried out periods of high flow of the Customers were taken into consideration, for this case, in a productive environment, the flow was started with 14,551 orders, which all came to complete the workflow satisfactorily. At the same time, emails sent to customers arrived effectively. Also, during the development of the test, 400 more orders were added, which did not diminish the performance. The tests were carried out in a period approx. 1 hour.

For the second iteration of stress tests, the IOPS of the RDS machine were increased to 4000 and the thread number was modified to 200 which allowed that when entering more than 400 orders every 3 minutes, the RDS maintained a constant balanced load of CPU With this modification, the peak of orders per minute is matched by presenting in the first iteration supporting the volumetric.

For each lambda function an IAM role is used where the "AWS Lambda Full Access", "AWS Lambda VPC Access Execution Role" and "oneClick_lambda_basic_execution_1515077751336" permission is granted. In addition, the duration of a session is controlled, and the date and time of the last activity that was taken.

If there are failed executions for example when you want to enter the queue reader and it fails in the 10 allowed attempts, it is housed in an SQS queue that is "waiting" and for these orders they must be pushed manually to go through the queue-reader and continue with the normal flow of the scheme until reach the accommodation in the database and the customer receives the email corresponding to the order processed.

For the solution "AWS Step Function" was used for the management of multiple services and in this case to manage the execution of two lambdas for when a failure occurs in the emission stage_DTE, so, the lambda is executed notifying the user that there was an incident with the order that will be processed.

For the designed scheme lambdas with VPC are not used because all the functions are within a public subnet.

The concurrence in which the lambdas can be executed, varies depending on it and for the process in which the order is found. It can vary from one hundred daily executions up to a thousand, depending on the lambda executed with a maximum duration of 11,900 milliseconds with the ratio of errors and satisfaction which usually has 100% satisfaction.

All the lambdas functions with which we work for our Customers are controlled with CloudWatch metrics such as invocations, durations, errors count and success rate, among other conditions.

AWS lambda is used for processing orders, for example, in the first step, a file is taken from an S3 bucket in format XML, which, is converted to json format and is sent to the SQS queue.

With lambda, the customer is given the processing of purchase orders stored in Amazon S3 buckets from when an order is entered until the order ends its flow, being stored in the PostgreSQL database.

Around 25 lambdas functions are used for the present solution, of which the most relevant are added to the diagram. All these are worked in the us-west-2 region of Oregon.

For each lambda function, executions per month may vary. Depending on the function, some reach 30,000 monthly executions approx., as there are others that can reach 190,000 approx., depending on the type of task you complete. All these executions are controlled with CloudWatch with control metrics and alarms, in addition to having constancy in CloudWatch logs.

The challenge that the Customers proposed to develop the solution was to improve the flow times of a purchase order, which was satisfactorily covered with AWS lambda and additional services, thus increasing the quality and improvements of the system development so that the calls the lambdas will take no more than 5 milliseconds. With this form of work, in which Amazon takes care of the servers, it allowed developers to improve the infrastructure and work of the application.

About the deployment & workload patterns and for the presented scheme the “all at once” implementation pattern is used with the eight lambdas functions presented so that all the work is updated and passes to the new version at the same time, without interrupting the flow of the orders. For the solution, the SAM (Serverless Application Model) implementation application model was used as an architecture framework due to its adaptability with other AWS services such as API, databases, etc., and therefore focus on the application and its development

Every lambda function connects to AWS services, including APIs, SQS services, database, SES mail service and AWS step functions, in addition to being all controlled with CloudWatch.

For the main processing and data flow of the orders within the implemented scheme, the lambdas functions are used due to their adaptation with the form of “Serverless” work without, with this, the developers and support focus on the development of the application and comply the requirements that the Customers requests. In addition, you can better manage the databases used in the solution, due to the feasibility of AWS managing the servers.

The system was designed based on the requirements requested by our Customers and based on the initial architecture provided by the architecture team, achieving a satisfactory work architecture, both for the Customers and for the development and support team.

ARCHITECTURE DIAGRAMS.