Web Analytics

Introduction

Website hosting on the cloud continues to grow in appeal as users understand the numerous benefits; scalability, simple and centralized management, and reduction of excess costs contribute to greater efficiency for website owners facing any level of network traffic. Website hosting on the cloud also enables access to a growing number of resources built specifically for handling large amounts of users and data without a decrease in performance. With the use of Event Hubs and Stream Analytics, another pathway exists for analyzing usage of a website deployed on the cloud, with the ability to respond to these results in real time. An application of this can be seen using Azure services to manage an e-commerce website and gather insights about clickstreams and general usage to provide the owner with valuable analytics-based insights and provide real-time product recommendations to the website users.

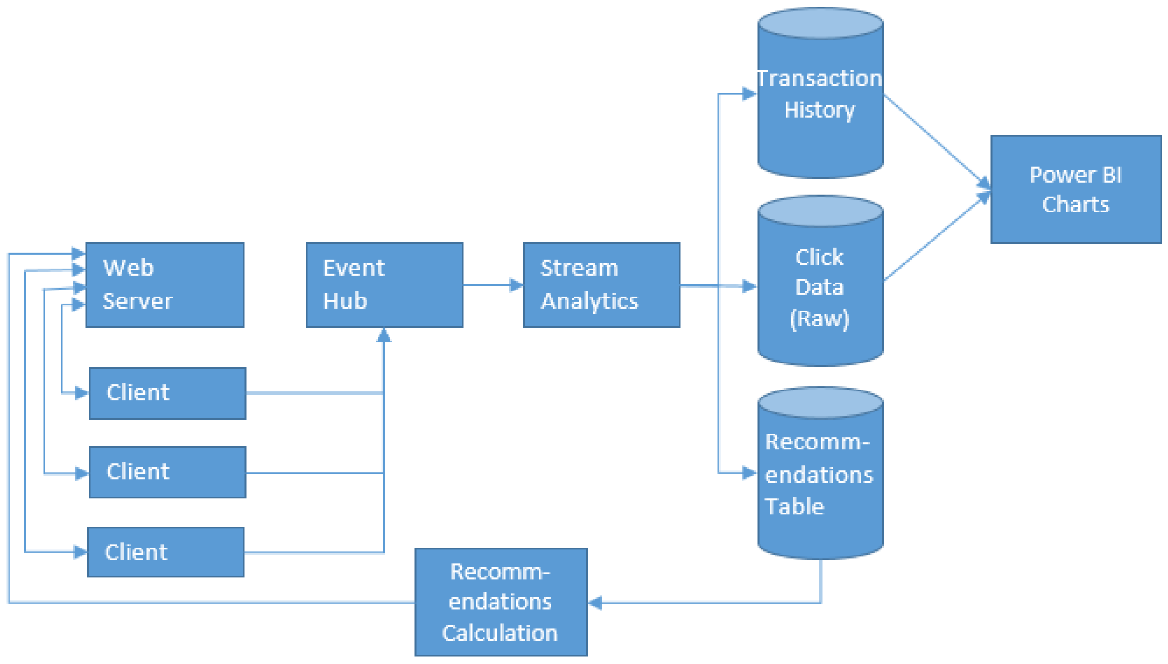
Enterprise Applications

Corporations have many incentives to host websites on the cloud, particularly the ability to pay only for used resources and to scale to larger consumer bases. However, these are not the only benefits. By managing a website using the Azure service portal, additional resources are made available to analyze and improve upon the existing site. With regard to e-commerce sites, consumer usage and purchase histories can be summarized for the website owner's viewing in real time. Trends and user data become immediately available for in-depth analysis.

By linking events occurring on the website, such as clicks and purchases, to an Event Hub as a data ingestion resource, a pipeline is initialized through which events can flow. After the events pass through the Event Hub, they can be sent to Stream Analytics for real-time processing using queries specified by the website owner. From this data, product recommendations can be determined by taking into account user browsing histories and past purchases. In addition, clickstream analysis can provide insight about which product links lead to the greatest number of views and how the site can be better organized.

These applications can be generalized for use on any web application deployed on Azure. Events generated on the website are ingested by an Event Hub then passed to Stream Analytics for real-time analysis. These results can then be stored in databases to be visualized using Power BI, integrated back into the website, or passed to another component of a more complex pipeline.

Pipeline



Data Collection and Storage

Events

Data will be generated as clicks and purchases made on the website, from which the fields of two classes of events can be populated.

Click Events:

- Event Type (1)
- Session ID
- User Email
- Previous URL
- Current URL
- Entry Timestamp
- Exit Timestamp

Purchase Events:

- Event Type (2)
- Transaction Number
- User Email
- Product
- Price
- Quantity
- Timestamp

When an event is generated by an action on the website, it can then be sent to an Event Hub using a JavaScript library (CryptoJS, built in serialization) to send events via HTTP POST from the web browser. After the event passes through the Event Hub, it is sent to a Stream Analytics job, processed, and sent to the appropriate database for further usage.

As shown in the Pipeline diagram above, raw data from purchase events and click events will be stored in databases to be later analyzed and visualized for the site owner through Power BI. This can be used to create a dashboard of usage data for the website, which the website owner can then use in whatever way is of interest. In our example solution, we look at user browsing data to suggest product recommendations and compute click-through rates of different pages and products.

Recommendations

There will be recommendations displayed on product pages as well as on the home page. The recommendations displayed on the product page will be specific to that product and will be the same for all users. Product recommendations displayed on the home page will be specific to the user.

Product page recommendations will consist of the items most commonly purchased with that product. Our recommendations database will store a list of each product and how many times each other product was bought with it. The table stores each item (x) and a list of every other item (y) along with a count of the number of times x was bought with y. The recommendations calculation finds the top 3 most similar products for a given product and displays them on the website.

A sample recommendation table is shown below for a website with 3 different products. Looking at row 1 as an example, we do not consider the count column with the same product ID. Overall transactions, product 2 was bought 3 times in the same transaction as product 1, and product 3 was bought 5 times in the same transaction as product 1. In this way, product 3 would be a better recommendation to display if the user is currently on the web page for product 1.

Table 1: Sample Recommendation Table for a Website with Three Products

Product Number	Count 1	Count 2	Count 3
1	-	3	5
2	3	-	4
3	5	4	-

Recommendations made on the home page will be based on past transactions if a specific user is logged in to an account and will be based on most popular products if the user is not logged into an account. The current season will also be considered when making recommendations on the home page.

In order to update recommendations each time a page is accessed or refreshed, we will use ASP.net SignalR to push content to the client website.

Clickstream Analysis

When analyzing click events, we will target certain key pieces of information to display to the website owner as an example of what can be done using Stream Analytics. Since the sample website is an e-commerce website, information of interest would deal with the effectiveness of the site's current organization and presentation. Breaking these categories into quantified metrics, we look at:

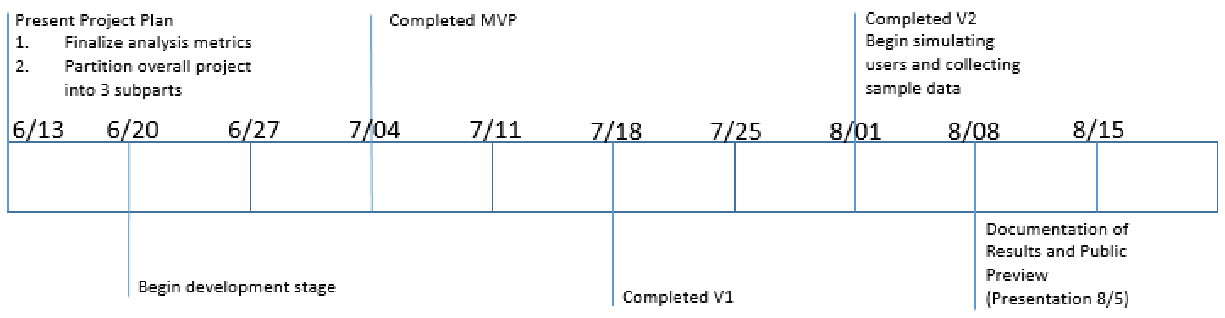
- Session Depth – How many product pages did the user go through before exiting the website or making a purchase?
- Average Page Browsing – How many pages were visited per session?
- Click-through Rates:
  - From the home page to a specific product page
  - From a product page to a different product page
  - From the home page to the shopping cart
  - From a product page to the shopping cart
- Average Viewing Time – How long did a user stay on a specific page?
- Returning Customers – What percentage of people that made one purchase returned to make another purchase?
- Purchase Rate – What percentage of browsing sessions ended in a purchase?

Using these metrics, the website owner can better determine how often users are viewing certain products, and what best leads to a purchase.

Simulating Load

We are going to simulate load using two methods. One method will simulate users on the site using VisualStudio.com's load testing feature. This will simulate users on the site. The other method we are going to use involves having a .NET client program send events directly to the event hub. This will allow us to test the back end part of our pipeline independently of the website portion. This will also allow specifying the event data that is sent regardless of how the website is programmed.

Development Timeline


Present Project Plan
1. Finalize analysis metrics
2. Partition overall project into 3 subparts
Completed MVP
Completed V2
Begin simulating users and collecting sample data
6/13
6/20
6/27
7/04
7/11
7/18
7/25
8/01
8/08
8/15
Begin development stage
Completed V1
Documentation of Results and Public Preview (Presentation 8/5)

MVP

- Setup Azure pipeline
- Setup events
- Simple data analysis (purchases per product using transaction history)
- Simulate large amounts of users to showcase Event Hub and Stream Analytics

V1 - First Iteration

- Click through rates and other analytics
- Optimize storage for SQL queries

V2 - Second Iteration

- Product Recommendations
- Add cart feature to the site

Later

- Real time data viewing on owner site
- Simplify deployment by using an ARM Template

Conclusion

Using an e-commerce website as an example, our solution showcases one way that multiple Azure services can be integrated into a scalable and easily maintainable web application with usage analytics.