**Scenario**

**Click analytics** is a special type of web analytics that gives special attention to user interactions with a web page: clicks (Point-and-click), mouse tracking, data entering, all the use actions that constitute the first stage in the conversion funnel. Commonly, click analytics focuses on on-site analytics. An editor of a web site uses click analytics to determine the performance of his or her particular site, with regards to where the users of the site are clicking. Also, click analytics may happen real-time or "unreal"-time, depending on the type of information sought. Typically, frontpage editors on high-traffic news media sites will want to monitor their pages in real-time, to optimize the content or understand user navigation patterns. Editors, designers or other types of stakeholders may analyze user actions on a wider time frame to aid them assess performance of writers, design elements or advertisements etc. Data about clicks may be gathered in at least two ways. Ideally, a click is "logged" when it occurs, and this method requires some functionality that picks up relevant information when the event occurs. Usually, a client-side script running in the web page collects and sends events to an ingestion pipeline that tracks and stores all the actions that happen during a user session. Once the user visit is complete, the data ingestion system sends a message to an hot path analytics system to process the session data. Alternatively, user sessions can be periodically analyzed by a cold path analytics system.

**Introduction**

This demo demonstrates how to build a highly scalable, highly reliable ingestion pipeline for a click analytics system that receives events sent by a client-side script running in a web page and persists them to an **Append Blob**, one for each user session. The demo uses a lightweight gateway service based on OWIN that receives events via the **HTTP** protocol by client-side scripts and sends them to an **Event Hub**. The use of an **Event Hub** allows to decouple the receive part of the ingestion pipeline from the persist operation (separation of concerns) and scale the two systems in a separate way. Another microservice is responsible for reading events from the Event Hub and persist them to an [Append Blob](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/), one for each user session. Since the processing of a user session starts only when the client-side script sends a special event to mark the end of the user visit and since page views inside a user sessions are processed in a sequential way, [Append Blobs](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/) are a good fit for storing user actions. [Append Blobs](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/) are similar to block blobs in that they are made up of blocks, but they are optimized for append operations, so they are useful for logging scenarios. A single block blob or append blob can contain up to 50,000 blocks of up to 4 MB each, for a total size of slightly more than 195 GB (4 MB X 50,000). For more information on Append Blobs, see [Understanding Block Blobs, Append Blobs, and Page Blobs](https://msdn.microsoft.com/library/azure/ee691964.aspx).

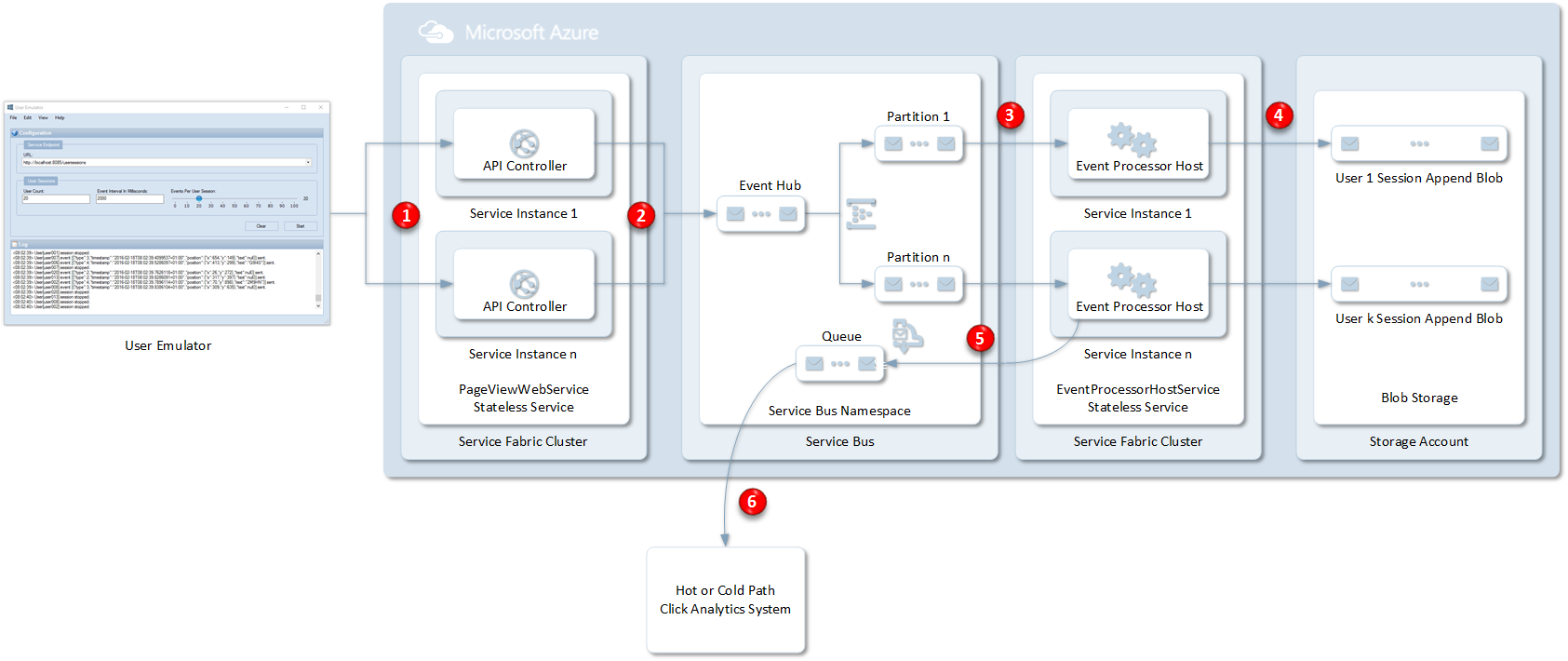
**License**

Microsoft Corporation (“Microsoft”) grants you a nonexclusive, perpetual, royalty-free right to use and modify the software code provided by us for the purposes of illustration  ("Sample Code") and to reproduce and distribute the object code form of the Sample Code, provided that you agree: (i) to not use our name, logo, or trademarks to market your software product in which the Sample Code is embedded; (ii) to include a valid copyright notice on your software product in which the Sample Code is embedded; and (iii) to indemnify, hold harmless, and defend us and our suppliers from and against any claims or lawsuits, whether in an action of contract, tort or otherwise, including attorneys’ fees, that arise or result from the use or distribution of the Sample Code or the use or other dealings in the Sample Code. Unless applicable law gives you more rights, Microsoft reserves all other rights not expressly granted herein, whether by implication, estoppel or otherwise.

THE SAMPLE CODE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL MICROSOFT OR ITS LICENSORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE SAMPLE CODE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

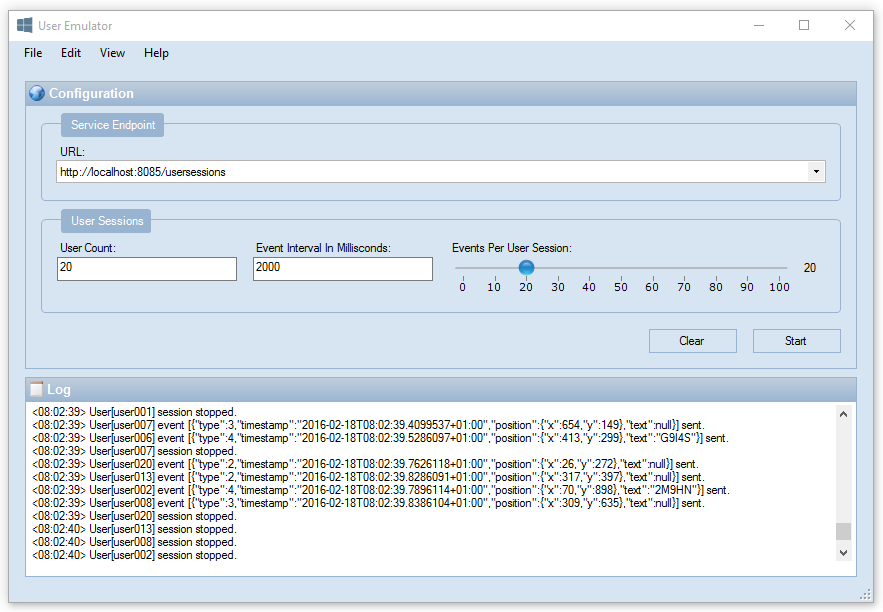
**Architecture Design**

The following picture shows the architecture design of the application.

[](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/Architecture.png?token=ABlOM93wLATEiZ9aC_OG45ZUgnYcwen5ks5W4o02wA%3D%3D)

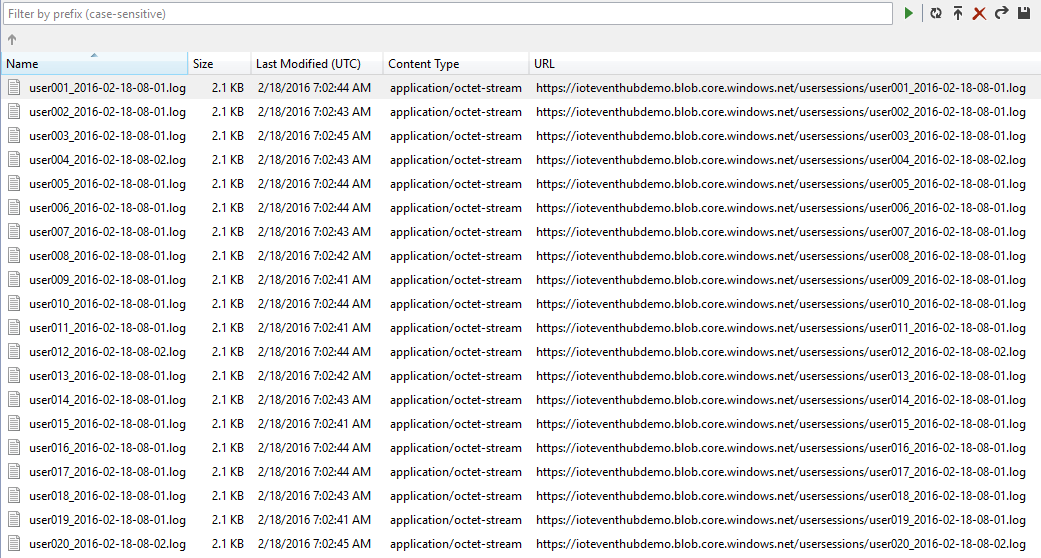
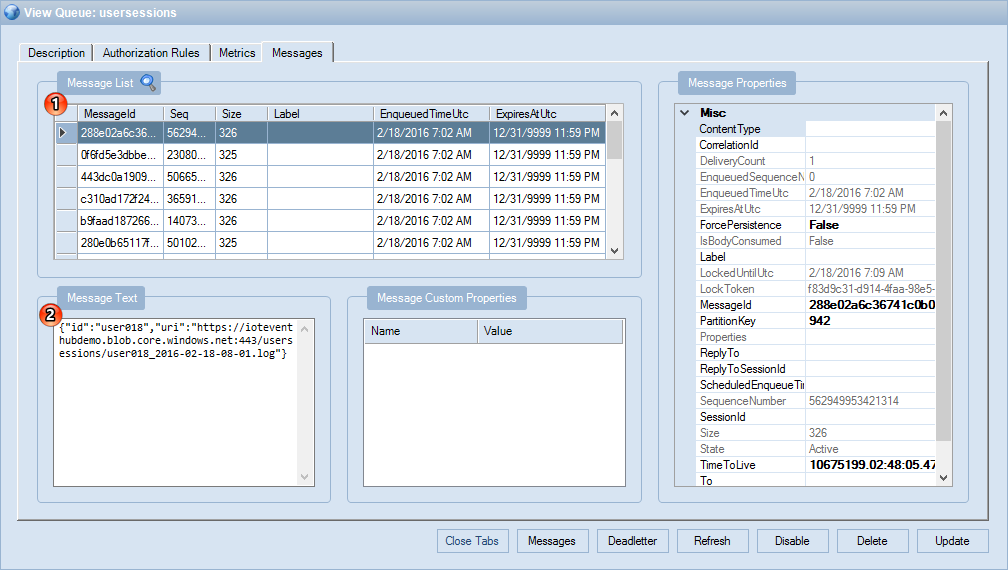
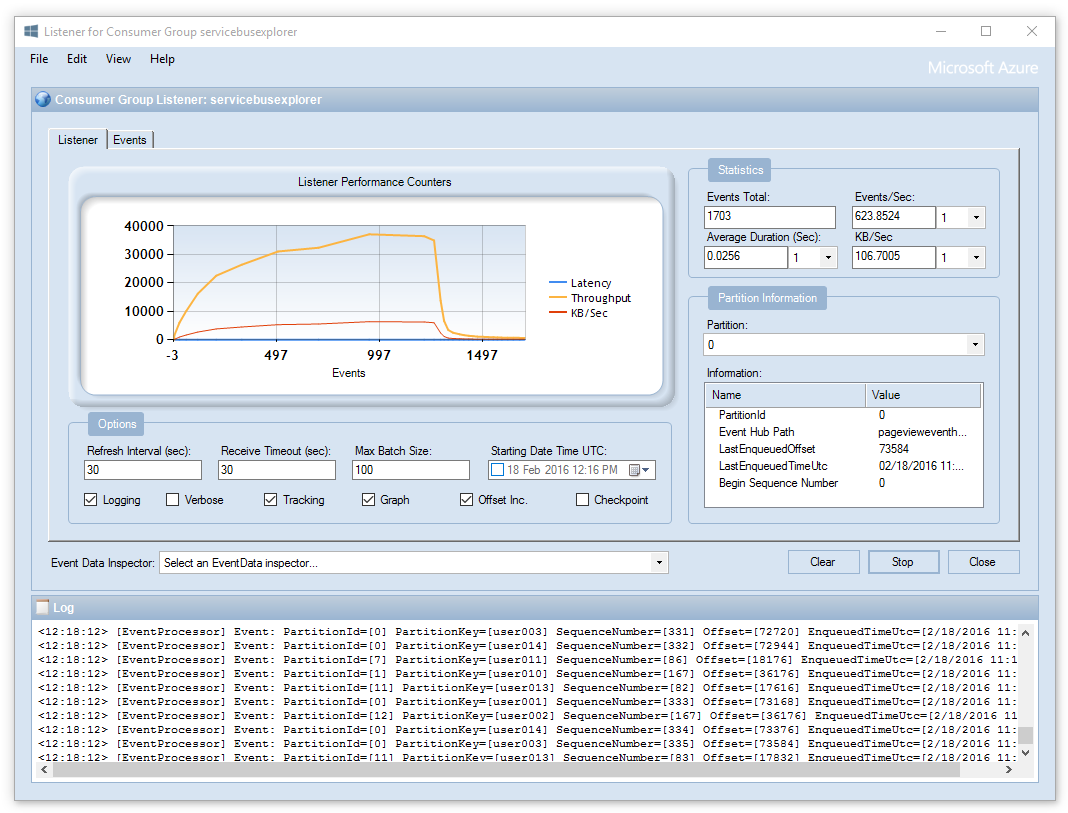
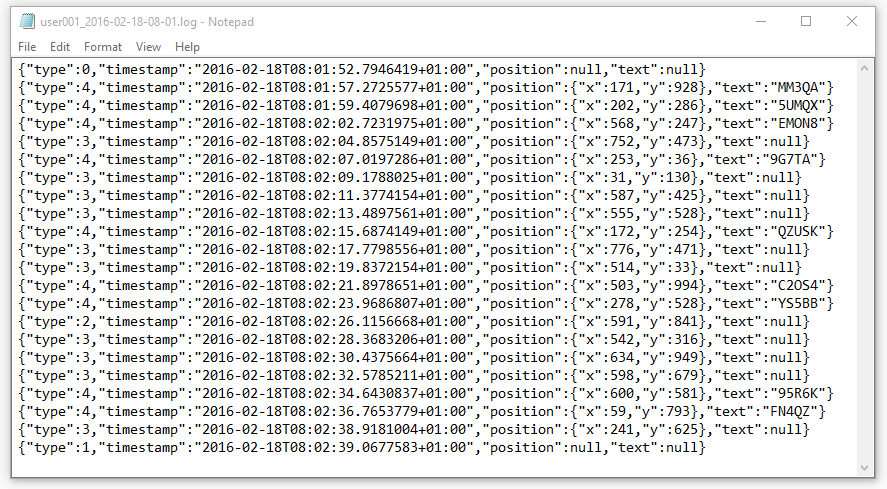
**Message Flow**

1. A Windows Forms application is used to emulate a configurable amount of users sending events to the ingestion pipeline of the click analytics system.

[](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/Client.png?token=ABlOM7xxg5AVFGKTQDZSX6Th2yFcwEskks5W4o1XwA%3D%3D)

The client application uses a separate Task to emulate each user. Each user session is composed by a series of JSON messages sent to the service endpoint of the click analytics ingestion pipeline:

* + a special session start event
  + a configurable amount of user events (click, mouse move, enter text)
  + a special session stop event

1. The **PageViewtWebService** stateless service receives requests using the **POST** method. The body of the request is in**JSON** format, the **Content-Type** header is equal to application/json, while the custom userId header contains the user id. The payload contains the **userId** (cross check) and the **User Event**. The service writes events into an **Event Hub**. The **userId** is used as a value for the [EventData.PartitionKey](https://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventdata.partitionkey.aspx) property. The **userid** is also stored in a custom **userId**property, while the **eventType** (start session, user event, stop session) is stored in another custom property of the[EventData](https://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventdata.aspx) message. Note: this microservice uses a pool of [EventHubClient](https://msdn.microsoft.com/library/azure/microsoft.servicebus.messaging.eventhubclient.aspx) objects to increase the throughput of the ingestion pipeline.
2. The **EventProcessorHostService** uses an **EventProcessorHost** listener to receive messages from the **Event Hub**.
3. The **EventProcessorHostService** retrieves the **userId** and **eventType** from the [Properties](https://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventdata.properties.aspx) collection of the [EventData](https://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventdata.aspx)message and the payload from the message body, and uses a [CloudAppendBlob](https://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storage.blob.cloudappendblob.aspx) object to write the event to a [Append Blob](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/) inside a given storage container.   
   The name is of the blob is **{userId}\_{session\_start\_timestamp}.log**.   
     
   [](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/Blobs.png?token=ABlOM1KMhtsUd_XUG3M0fM8ria8Vm10lks5W4o1swA%3D%3D)  
     
   When the it receives a stop session event, the microservice sends a **JSON** message to **Service Bus Queue**. The message contains the **userId** and **uri** of the [Append Blob](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/) containing the events of the user visit. The message is received and processed by an external hot path analytics system. You can use the [Service Bus Explorer](https://github.com/paolosalvatori/ServiceBusExplorer) to read messages from the **Service Bus Queue**, as shown in the following picture.   
     
   [](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/QueueMessage.png?token=ABlOM_WG5Ibe-wkMg5NbHnTMILexnyOuks5W4o17wA%3D%3D)  
     
   To monitor the message flow in real-time, you can create a test **Consumer Group** other than the one used by the application, and use the aaaaaaaa to create and run a **Consumer Group Listener**, as shown in the following picture.  
   [](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/EventHub.png?token=ABlOM7p7_zO5T_ISzG1UA71biSUyJFS0ks5W4o2YwA%3D%3D)  
     
   Each **Append Blob** contains all the user events in **JSON** format tracked during the user session:   
     
   [](https://raw.githubusercontent.com/paolosalvatori/PageViewTracer/master/Images/BlobContent.png?token=ABlOM9Yt5yUP2_kNIZTPNEoYtGunAfBrks5W4o3QwA%3D%3D)

**Service Fabric Application**

The Service Fabric application ingest events from the input Event Hub, processes sensor readings and generates an alert whenever a value outside of the tolerance range is received. The application is composed of three services:

* **PageViewWebService**: this is a stateless service hosting **OWIN** and exposing a REST ingestion service. The service has been implemented using an **ASP.NET Web API** REST service. The service is implemented as an **ApiController**that exposes a **POST** method invoked by client-side scripts. The service uses a pool of **EventHubClient** objects to increase the performance. Each **EventHubClient** object is cached in a static list and uses an **AMQP** session to send events into the **Event Hub**.
* **EventProcessorHostService**: this is a stateless service that creates an **EventProcessorHost** listener to receive messages from the event hub. Note: to maximize the throughput, make sure that the number of service instances and cluster nodes matches the number of event hub partitions. The **ProcessEventsAsync** method of the **EventProcessor**class creates and caches a [CloudAppendBlob](https://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storage.blob.cloudappendblob.aspx) object to write the event to a [Append Blob](https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/) object, for each user session, to write page views to append blobs.

**Note**: one of the advantages of stateless services over stateful services is that by specifying **InstanceCount="-1"** in the**ApplicationManifest.xml**, you can create an instance of the service on each node of the Service Fabric cluster. When the cluster uses [Virtual Machine Scale Sets](https://azure.microsoft.com/en-gb/documentation/articles/virtual-machines-vmss-overview/) to to scale up and down the number of cluster nodes, this allows to automatically scale up and scale down the number of instances of a stateless service based on the autoscaling rules and traffic conditions.

**Recommendations**

In order to maximize the throughput of the demo, put in practice the following recommendations:

* Deploy the **Service Fabric** application on a cluster with at least 16 nodes
* Make sure to create an **Event Hub** with a sufficient number of partitions (for example, 16) in a dedicated Service Bus namespace with no other **Event Hubs** and increase the number to **Throughput Units** to be equal to the number of partitions of the **Event Hub**, at least for the duration of the load tests
* Modify the code to write **Append Blobs** to a pool of **Storage Accounts** instead of a single **Storage Account** as in the current implementation
* Deploy the **Service Fabric** services with the **InstanceCount** attribute equal to -1
* Repeat the test when the **Service Fabric** cluster will support **Virtual Machines Scale Sets**
* Make sure to properly configure **Visual Studio Load Test** to generate enough traffic against the Azure-hosted application. Consider using multiple instances of the Load Test running on multiple Azure subscriptions.

**Application Configuration**

Make sure to replace the following placeholders in the project files below before deploying and testing the application on the local development Service Fabric cluster or before deploying the application to your Service Fabric cluster on Microsoft Azure.

**Placeholders**

This list contains the placeholders that need to be replaced before deploying and running the application:

* **[ServiceBusConnectionString]**: defines the connection string of the **Service Bus** namespace that contains the **Event Hub** and **Queue** used by the solution.
* **[StorageAccountConnectionString]**: contains the connection string of the **Storage Account** used by the**EventProcessorHost** to store partition lease information when reading data from the input **Event Hub**.
* **[EventHubName]**: contains the name of the input **Event Hub**.
* **[ConsumerGroupName]**: contains the name of the **Consumer Group** used by the **EventProcessorHost** to read data from the input **Event Hub**.
* **[ContainerName]**: defines the name of the **Storage Container** where the **EventProcessorHost** writes **Append Blobs**.
* **[QueueName]**: contains the name of the **Service Bus Queue** used by the **EventProcessorHost** send a message to the external hot path analytics system when user session completes.
* **[CheckpointCount]**: this number defines after how many messages the **EventProcessorHost** invokes the**ChechpointAsync** method.
* **[EventHubClientNumber]**: this number defines how many [EventHubClient](https://msdn.microsoft.com/library/azure/microsoft.servicebus.messaging.eventhubclient.aspx) objects are contained in the connection pool of the **PageViewWebService**.

**Configuration Files**

**App.config** file in the **UserEmulator** project:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<appSettings>

<add key="url" value="http://localhost:8085/usersessions;http://[SF\_CLUSTER\_NAME].[REGION].cloudapp.azure.com:8085/usersessions"/>

<add key="userCount" value="20"/>

<add key="eventInterval" value="2000"/>

<add key="eventsPerUserSession" value="50"/>

</appSettings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>

</startup>

<runtime>

<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>

<assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral"/>

<bindingRedirect oldVersion="0.0.0.0-8.0.0.0" newVersion="8.0.0.0"/>

</dependentAssembly>

</assemblyBinding>

</runtime>

</configuration>

**ApplicationParameters\Local.xml** file in the **PageViewTracer** project:

<?xml version="1.0" encoding="utf-8"?>

<Application xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Name="fabric:/PageViewTracer" xmlns="http://schemas.microsoft.com/2011/01/fabric">

<Parameters>

<Parameter Name="EventProcessorHostService\_InstanceCount" Value="1" />

<Parameter Name="EventProcessorHostService\_StorageAccountConnectionString" Value="[StorageAccountConnectionString]" />

<Parameter Name="EventProcessorHostService\_ServiceBusConnectionString" Value="[ServiceBusConnectionString]" />

<Parameter Name="EventProcessorHostService\_EventHubName" Value="[EventHubName]" />

<Parameter Name="EventProcessorHostService\_ConsumerGroupName" Value="[ConsumerGroupName]" />

<Parameter Name="EventProcessorHostService\_ContainerName" Value="[ContainerName]" />

<Parameter Name="EventProcessorHostService\_QueueName" Value="[QueueName]" />

<Parameter Name="EventProcessorHostService\_MaxRetryCount" Value="3" />

<Parameter Name="EventProcessorHostService\_CheckpointCount" Value="[CheckpointCount]" />

<Parameter Name="EventProcessorHostService\_BackoffDelay" Value="1" />

<Parameter Name="PageViewWebService\_InstanceCount" Value="1" />

<Parameter Name="PageViewWebService\_ServiceBusConnectionString" Value="[ServiceBusConnectionString]" />

<Parameter Name="PageViewWebService\_EventHubName" Value="[EventHubName]" />

<Parameter Name="PageViewWebService\_EventHubClientNumber" Value="[EventHubClientNumber]" />

<Parameter Name="PageViewWebService\_MaxRetryCount" Value="3" />

<Parameter Name="PageViewWebService\_BackoffDelay" Value="1" />

</Parameters>

</Application>

**ApplicationParameters\Cloud.xml** file in the **PageViewTracer** project:

<?xml version="1.0" encoding="utf-8"?>

<Application xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Name="fabric:/PageViewTracer" xmlns="http://schemas.microsoft.com/2011/01/fabric">

<Parameters>

<Parameter Name="EventProcessorHostService\_InstanceCount" Value="-1" />

<Parameter Name="EventProcessorHostService\_StorageAccountConnectionString" Value="[StorageAccountConnectionString]" />

<Parameter Name="EventProcessorHostService\_ServiceBusConnectionString" Value="[ServiceBusConnectionString]" />

<Parameter Name="EventProcessorHostService\_EventHubName" Value="[EventHubName]" />

<Parameter Name="EventProcessorHostService\_ConsumerGroupName" Value="[ConsumerGroupName]" />

<Parameter Name="EventProcessorHostService\_ContainerName" Value="[ContainerName]" />

<Parameter Name="EventProcessorHostService\_QueueName" Value="[QueueName]" />

<Parameter Name="EventProcessorHostService\_MaxRetryCount" Value="3" />

<Parameter Name="EventProcessorHostService\_CheckpointCount" Value="[CheckpointCount]" />

<Parameter Name="EventProcessorHostService\_BackoffDelay" Value="1" />

<Parameter Name="PageViewWebService\_InstanceCount" Value="-1" />

<Parameter Name="PageViewWebService\_ServiceBusConnectionString" Value="[ServiceBusConnectionString]" />

<Parameter Name="PageViewWebService\_EventHubName" Value="[EventHubName]" />

<Parameter Name="PageViewWebService\_EventHubClientNumber" Value="[EventHubClientNumber]" />

<Parameter Name="PageViewWebService\_MaxRetryCount" Value="3" />

<Parameter Name="PageViewWebService\_BackoffDelay" Value="1" />

</Parameters>

</Application>

**ApplicationManifest.xml** file in the **PageViewTracer** project:

<?xml version="1.0" encoding="utf-8"?>

<ApplicationManifest xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ApplicationTypeName="PageViewTracerType" ApplicationTypeVersion="1.0.1" xmlns="http://schemas.microsoft.com/2011/01/fabric">

<Parameters>

<Parameter Name="EventProcessorHostService\_InstanceCount" DefaultValue="-1" />

<Parameter Name="EventProcessorHostService\_StorageAccountConnectionString" DefaultValue="" />

<Parameter Name="EventProcessorHostService\_ServiceBusConnectionString" DefaultValue="" />

<Parameter Name="EventProcessorHostService\_EventHubName" DefaultValue="" />

<Parameter Name="EventProcessorHostService\_ConsumerGroupName" DefaultValue="" />

<Parameter Name="EventProcessorHostService\_ContainerName" DefaultValue="usersessions" />

<Parameter Name="EventProcessorHostService\_QueueName" DefaultValue="usersessions" />

<Parameter Name="EventProcessorHostService\_MaxRetryCount" DefaultValue="3" />

<Parameter Name="EventProcessorHostService\_CheckpointCount" DefaultValue="100" />

<Parameter Name="EventProcessorHostService\_BackoffDelay" DefaultValue="1" />

<Parameter Name="PageViewWebService\_InstanceCount" DefaultValue="-1" />

<Parameter Name="PageViewWebService\_ServiceBusConnectionString" DefaultValue="" />

<Parameter Name="PageViewWebService\_EventHubName" DefaultValue="" />

<Parameter Name="PageViewWebService\_EventHubClientNumber" DefaultValue="32" />

<Parameter Name="PageViewWebService\_MaxRetryCount" DefaultValue="3" />

<Parameter Name="PageViewWebService\_BackoffDelay" DefaultValue="1" />

</Parameters>

<ServiceManifestImport>

<ServiceManifestRef ServiceManifestName="EventProcessorHostServicePkg" ServiceManifestVersion="1.0.0" />

<ConfigOverrides>

<ConfigOverride Name="Config">

<Settings>

<Section Name="EventProcessorHostConfig">

<Parameter Name="StorageAccountConnectionString" Value="[EventProcessorHostService\_StorageAccountConnectionString]" />

<Parameter Name="ServiceBusConnectionString" Value="[EventProcessorHostService\_ServiceBusConnectionString]" />

<Parameter Name="EventHubName" Value="[EventProcessorHostService\_EventHubName]" />

<Parameter Name="ConsumerGroupName" Value="[EventProcessorHostService\_ConsumerGroupName]" />

<Parameter Name="ContainerName" Value="[EventProcessorHostService\_ContainerName]" />

<Parameter Name="QueueName" Value="[EventProcessorHostService\_QueueName]" />

<Parameter Name="CheckpointCount" Value="[EventProcessorHostService\_CheckpointCount]" />

<Parameter Name="MaxRetryCount" Value="[EventProcessorHostService\_MaxRetryCount]" />

<Parameter Name="BackoffDelay" Value="[EventProcessorHostService\_BackoffDelay]" />

</Section>

</Settings>

</ConfigOverride>

</ConfigOverrides>

</ServiceManifestImport>

<ServiceManifestImport>

<ServiceManifestRef ServiceManifestName="PageViewWebServicePkg" ServiceManifestVersion="1.0.1" />

<ConfigOverrides>

<ConfigOverride Name="Config">

<Settings>

<Section Name="PageViewWebServiceConfig">

<Parameter Name="ServiceBusConnectionString" Value="[PageViewWebService\_ServiceBusConnectionString]" />

<Parameter Name="EventHubName" Value="[PageViewWebService\_EventHubName]" />

<Parameter Name="EventHubClientNumber" Value="[PageViewWebService\_EventHubClientNumber]" />

<Parameter Name="MaxRetryCount" Value="[PageViewWebService\_MaxRetryCount]" />

<Parameter Name="BackoffDelay" Value="[PageViewWebService\_BackoffDelay]" />

</Section>

</Settings>

</ConfigOverride>

</ConfigOverrides>

</ServiceManifestImport>

<DefaultServices>

<Service Name="EventProcessorHostService">

<StatelessService ServiceTypeName="EventProcessorHostServiceType" InstanceCount="[EventProcessorHostService\_InstanceCount]">

<SingletonPartition />

</StatelessService>

</Service>

<Service Name="PageViewWebService">

<StatelessService ServiceTypeName="PageViewWebServiceType" InstanceCount="[PageViewWebService\_InstanceCount]">

<SingletonPartition />

</StatelessService>

</Service>

</DefaultServices>

</ApplicationManifest>