Data Analysis using Hadoop: Module 6, Lesson 5  
Getting Started with Event Hubs Hands-On Lab

## Overview

In this lab, you will create an Event Hubs in Azure, and send message to Event Hubs using sender program written by C language.

## Objectives

In this hands-on lab you will learn how to:

* How to provision the Event Hub in Azure portal
* How to send messages to Event Hubs

## Prerequisites

The following are required to complete this hands-on lab:

* A Microsoft Azure subscription
* A C development environment – gcc stack on Linux VM with Ubuntu 14.04
* Microsoft Visual Studio Express for Windows

Note : The Azure portal is continually improved and changed. The steps in this exercise reflect the user interface of the Microsoft Azure portal at the time of writing, but may not match the latest design of portal.

## Exercises

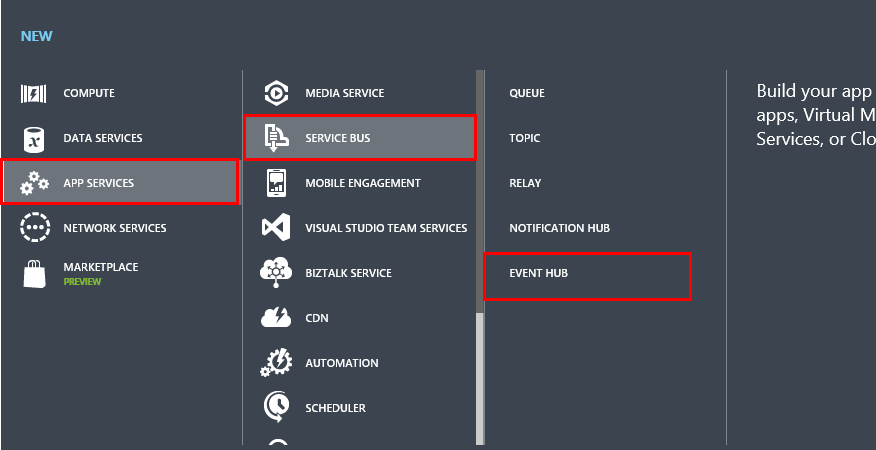
This hands-on lab includes the following exercises:

* Exercise 1: Create an Event Hub
* Exercise 2: Send message to Event Hubs

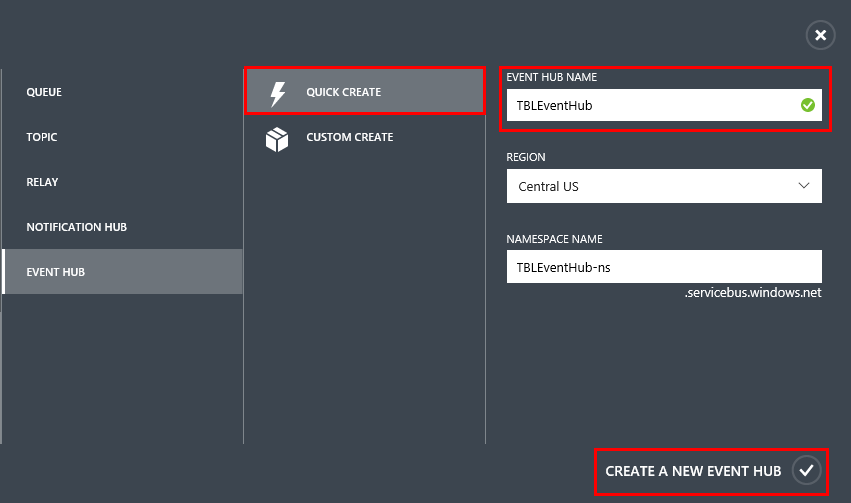
## Exercise 1: Create an Event Hub

The first task you have to perform is to creation of Event Hub.

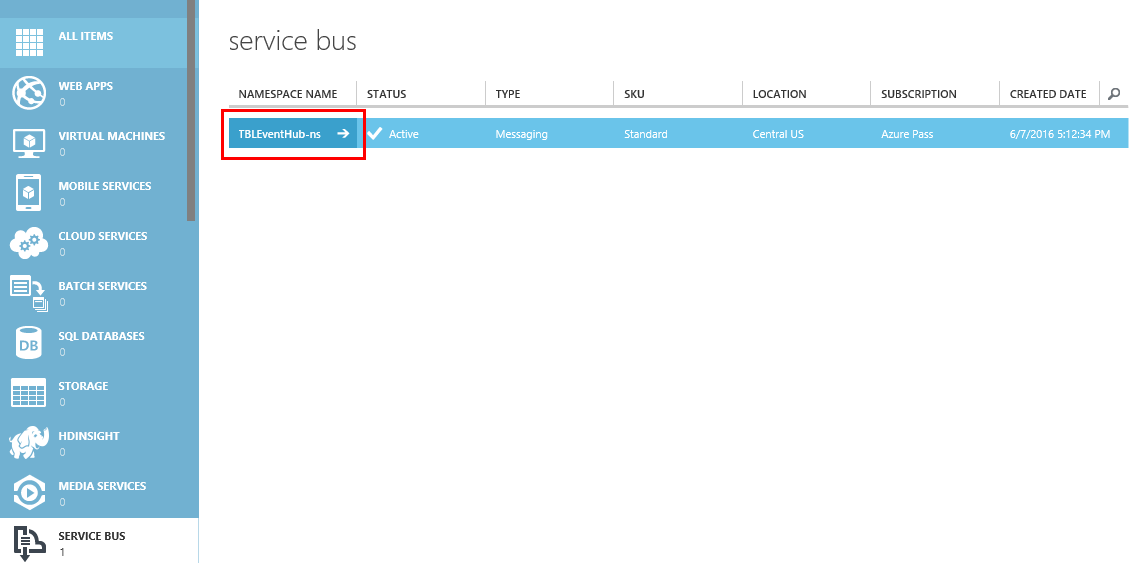
1. In a web browser, navigate to Azure classic portal, https://manage.windowsazure.com/. Sign into the portal using your subscription, and click NEW at the bottom to create an Event Hub.

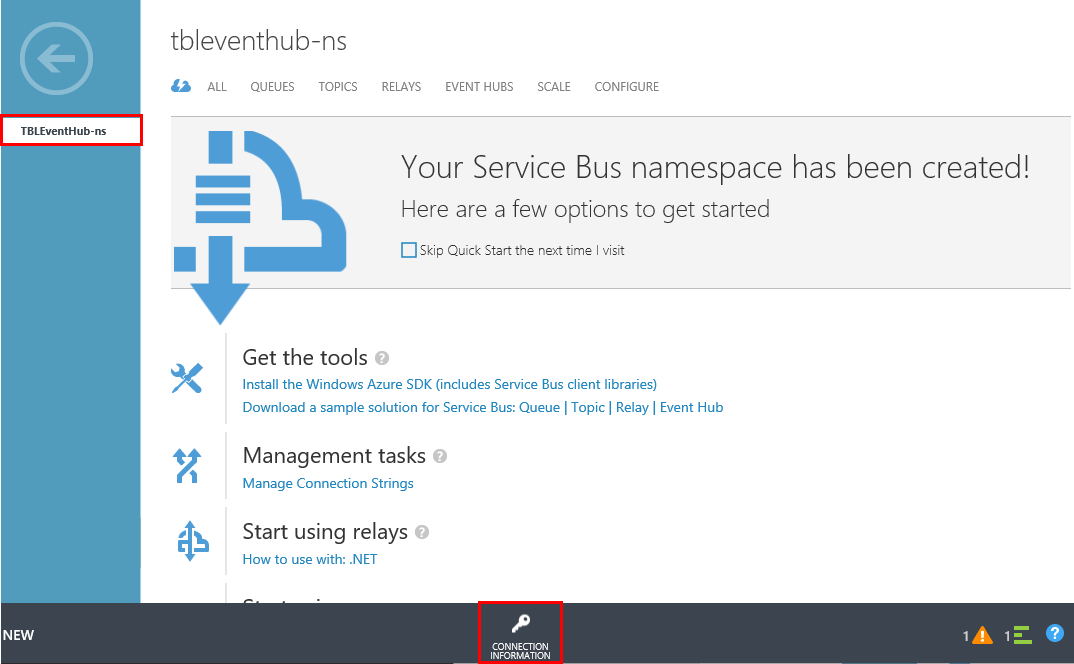


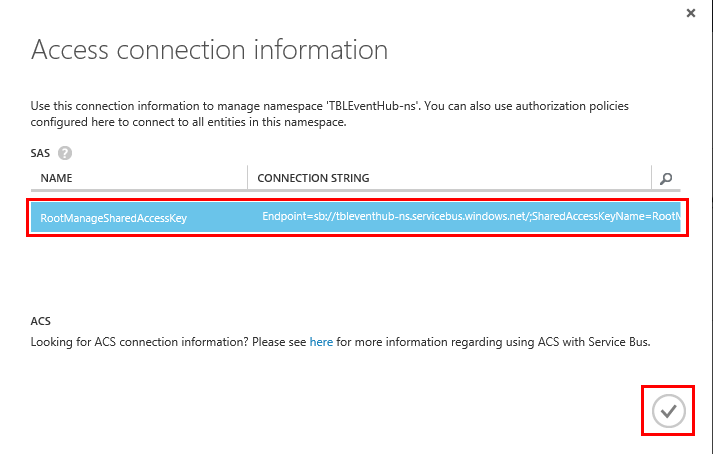
1. Click on App Service, then Service Bus, then Event Hub and Quick Create. This quickly create an Event Hub with default configuration settings.
2. Enter a name for your Event Hub, select your desired region, and the click “Create a new Event Hub”



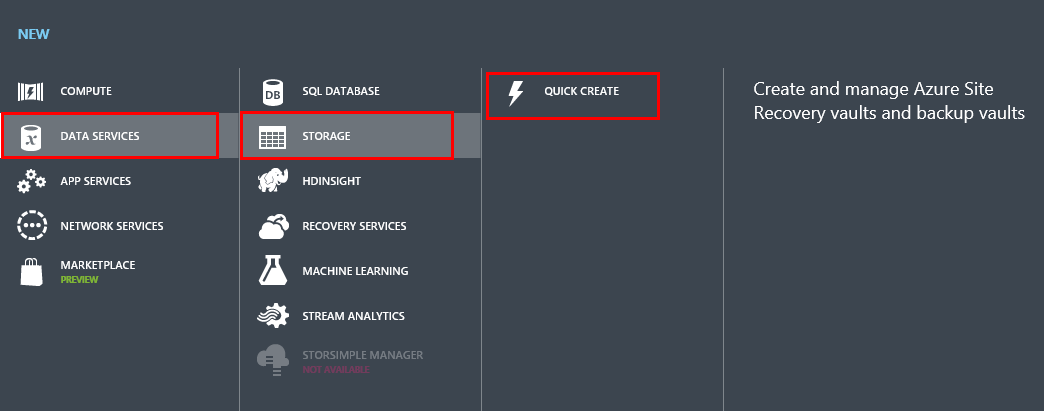
1. The portal creates a namespace for your Event Hub. Click this namespace (in this lab, TBLEventHub-ns).



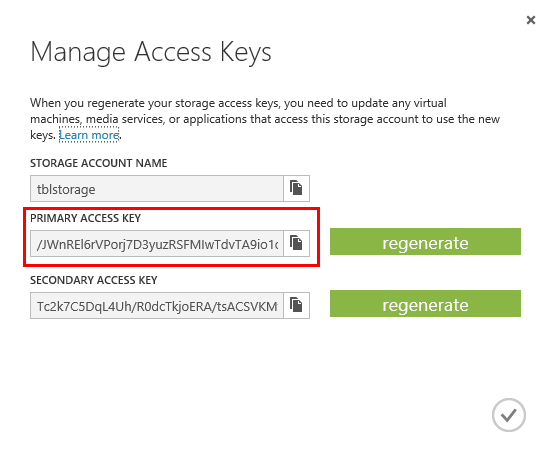
1. When the Event Hub has been created successfully, the Bus namespace also has been created. At the bottom of the page, click “Connection Information”. 
2. Show the Access connection information for Event Hub. Click the copy button to copy the RootManageSharedAccessKey connection string to the clipboard. Save this information to use later.



1. Now, the user should have the connection strings for both send and receive events. And make an Azure Storage account for managing events. Click NEW at the bottom of page, and click “Data Service, then Storage, then “Quick Create”, and then type a name for your storage account. Select your region, and then click “Create Storage Account”



1. Click the newly created storage account, and then click “Manage Access Keys”. Copy the primary access key to use later in this lab.



## Exercise 2: Send messages to Event Hubs.

The second task you have to perform is to write a C app to send events to your Event Hub.

1. First, build an environment that can make the sender program as follows;
   1. Create a new VM - Ubuntu Server 14.04 LTS. If necessary, use putty program to connect to your Linux VM.
   2. Proton AMQP library from Apache Qpid project.
2. To compile the Proton library, install the following packages:
   1. $ sudo apt-get install build-essential cmake uuid-dev openssl libssl-dev
   2. $ wget http://archive.apache.org/dist/qpid/proton/0.7/qpid-proton-0.7.tar.gz
   3. $ tar xvfz qpid-proton-0.7.tar.gz
   4. $ cd qpid-proton-0.7
   5. $ mkdir build
   6. $ cd build
   7. $ cmake -DCMAKE\_INSTALL\_PREFIX=/usr ..
   8. $ sudo make install
3. In your working directory, create a new file, sender.c. Remember to substitute the value of your Event Hub name and namespace name using your Event Hub information from exercise 1.
4. Compile with gcc.
5. C code for sender.c

#include "proton/message.h"

#include "proton/messenger.h"

#include <getopt.h>

#include <proton/util.h>

#include <sys/time.h>

#include <stddef.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

#define check(messenger) \

{ \

if(pn\_messenger\_errno(messenger)) \

{ \

printf("check\n"); \

die(\_\_FILE\_\_, \_\_LINE\_\_, pn\_error\_text(pn\_messenger\_error(messenger))); \

} \

}

pn\_timestamp\_t time\_now(void)

{

struct timeval now;

if (gettimeofday(&now, NULL)) pn\_fatal("gettimeofday failed\n");

return ((pn\_timestamp\_t)now.tv\_sec) \* 1000 + (now.tv\_usec / 1000);

}

void die(const char \*file, int line, const char \*message)

{

printf("Dead\n");

fprintf(stderr, "%s:%i: %s\n", file, line, message);

exit(1);

}

int sendMessage(pn\_messenger\_t \* messenger) {

char \* address = (char \*) "amqps://SendRule:{Send Rule key}@{namespace name}.servicebus.windows.net/{event hub name}";

char \* msgtext = (char \*) "Hello from C!";

pn\_message\_t \* message;

pn\_data\_t \* body;

message = pn\_message();

pn\_message\_set\_address(message, address);

pn\_message\_set\_content\_type(message, (char\*) "application/octect-stream");

pn\_message\_set\_inferred(message, true);

body = pn\_message\_body(message);

pn\_data\_put\_binary(body, pn\_bytes(strlen(msgtext), msgtext));

pn\_messenger\_put(messenger, message);

check(messenger);

pn\_messenger\_send(messenger, 1);

check(messenger);

pn\_message\_free(message);

}

int main(int argc, char\*\* argv) {

printf("Press Ctrl-C to stop the sender process\n");

pn\_messenger\_t \*messenger = pn\_messenger(NULL);

pn\_messenger\_set\_outgoing\_window(messenger, 1);

pn\_messenger\_start(messenger);

while(true) {

sendMessage(messenger);

printf("Sent message\n");

sleep(1);

}

// release messenger resources

pn\_messenger\_stop(messenger);

pn\_messenger\_free(messenger);

return 0;

}

## Summary

In this hands-on lab, you learned how to:

* How to provision the Event Hub in Azure portal
* How to send messages to Event Hubs