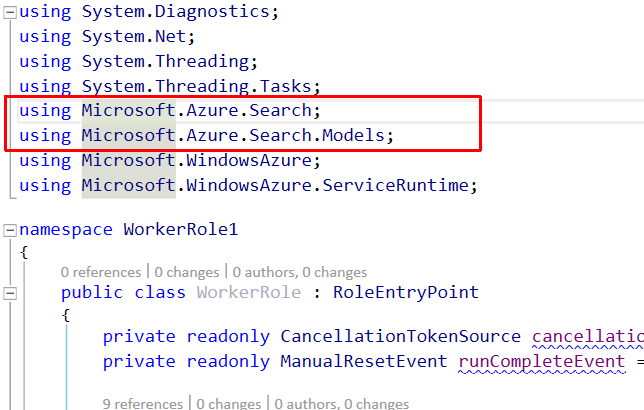
Azure Search

1. Create new search index
2. Fetch new tweets from the queue
3. Upload new tweets to the index
4. Search the index

# Part 1: Create new search index

1. Add new project -> Cloud -> Azure Cloud Serivce -> Worker Role
2. Add NuGet package “Microsoft.Azure.Search” -> 0.13.0-preview



1. Add configuration for Search API in the cloud service (“SearchServiceName”) and (“SearchApiKey”)
2. Fetch the cloud configuration settings OnStart()

var searchServiceName = CloudConfigurationManager.GetSetting("SearchServiceName");

var apiKey = CloudConfigurationManager.GetSetting("SearchApiKey");

1. Create the search client

var searchServiceClient = new SearchServiceClient(searchServiceName, new SearchCredentials(apiKey));

1. Define an index for tweet data

var index = new Index()

{

    Name = "tuttweets",

    Fields = new[]

    {

        new Field("tweetId", DataType.String) {IsKey=true},

        new Field("username", DataType.String) {IsSearchable=true, IsFilterable=true},

        new Field("tweetMessage", DataType.String) {IsSearchable=true, IsFilterable=true},

        new Field("createdAt", DataType.DateTimeOffset) {IsSortable=true, IsFilterable=true}

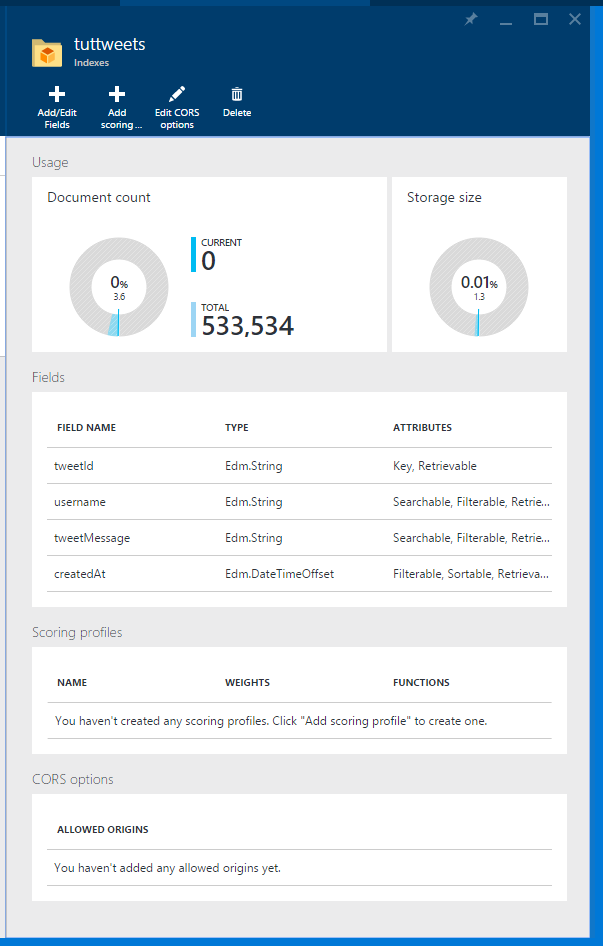
    }

};

1. Upload the index to Azure Search:

searchServiceClient.Indexes.CreateOrUpdate(index);

1. Start the cloud service
2. Inspect the search index Azure Portal:



Completed code:



# Part 2: Fetch new tweets from the queue

1. Create a CloudQueue field:

private CloudQueue \_searchIndexQueue;

1. Connect to the storage account with the queue from tweet event publisher OnStart()

var storageAccount = CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));

var queueClient = storageAccount.CreateCloudQueueClient();

\_searchIndexQueue = queueClient.GetQueueReference("searchindexqueue");

\_searchIndexQueue.CreateIfNotExists();

1. In RunAsync(), fetch tweets from the queue

var tweets = new List<Tweet>();

foreach (var msg in \_searchIndexQueue.GetMessages(10).Where(msg => msg != null))

{

    var messageString = msg.AsString;

    tweets.Add(JsonConvert.DeserializeObject<Tweet>(messageString));

    \_searchIndexQueue.DeleteMessage(msg);

}

1. Make sure TweetEventPublisher and TweetEventHandler are setup to run
2. Run your worker role and see if you found any tweets

Trace.TraceInformation("WorkerRole1 fetched {0} tweets", tweets.Count);

# Part 3: Upload new tweets to the index

1. Get a SearchIndexClient for the named Index you created above

var client = \_searchServiceClient.Indexes.GetClient("tuttweets");

1. Create a flat object of tweet data you can upload to the index

var flatTweets = tweets.Select(tweet => new FlattendTweet

{

    TweetId = tweet.IdString,

    Username = tweet.User.ScreenName,

    CreatedAt = tweet.CreatedAtUtc,

    TweetMessage = tweet.Text,

}).ToList();

1. Define a list of Index Actions for the flat tweets

var actions = new List<IndexAction<FlattendTweet>>();

foreach (var t in flatTweeets)

{

    var action = new IndexAction<FlattendTweet>(IndexActionType.**Upload**, t);

    actions.Add(action);

}

1. Add the actions to a batch, and Index them

var batch = new IndexBatch<FlattendTweet>(actions);

client.Documents.Index(batch);

1. We can simplifiy it, since the default IndexActionType is Upload

client.Documents.Index(IndexBatch.Create(flatTweeets.Select(IndexAction.Create)));

# Part 4: Add a search view

1. Add new project -> Web -> ASP.NET Web Application -> ASP.NET 5 Preview Templates -> Web Application
2. Add Microsoft.Azure.Search and Entities.Package as dependencies in project.json

"frameworks": {

  "dnx451": {

    "dependencies": {

      "Microsoft.Azure.Search": "0.13.0-preview",

      "Entities.Package": ""

    }

  }

},

1. Set up configuration and client for search index in HomeController.cs

private static readonly string SearchServiceName = "tjsearch";

private static readonly string ApiKey = "C10F3A0A73E30B5AB9483413EF846E97";

private static readonly SearchServiceClient ServiceClient = new SearchServiceClient

(SearchServiceName, new SearchCredentials(ApiKey));

private readonly SearchIndexClient \_indexClient = ServiceClient.Indexes.GetClient

("tuttweets");

1. Add a search query using the SearchIndexClient in the Index action

var searchParameters = new SearchParameters

{

    OrderBy = new List<string> { "CreatedAt" },

    IncludeTotalResultCount = true

};

var response = \_indexClient.Documents.Search<FlattendTweet>(“\*”, searchParameters);

1. Return the response to the View

return View(response);

1. Add the DocumentSearchResponse as a model in view Index.cshtml

@using System.Globalization

@using Microsoft.Azure.Search.Models

@model DocumentSearchResponse<Entities.Twitter.SearchIndex.FlattendTweet>

1. Display the search results in the view

<div class="row">

    <div class="col-md-12">

        <h1>Tweets returned: @Model.Results.Count</h1>

        @foreach (var tweet in @Model.Results)

        {

            <p>@tweet.Document</p>

        }

    </div>

</div>

1. Create a search form in the View

<div class="row">

    <div class="col-md-12">

        <form action="Home\Index" method="get" id="search-form">

            <div class="input-group col-md-5">

                <input type="search" name="query" id="query" value="@ViewBag.searchString" autocomplete="off" class="form-control input-lg">

                <span class="input-group-btn">

                        <button class="btn btn-primary btn-lg" type="submit">Search</button>

                    </span>

            </div>

        </form>

    </div>

</div>

1. Add the query as a parameter to the Index action

public IActionResult Index(string query = "\*")

## Result

Controller code:



View code:

