**Lesson13 Deploy Azure Batch Service Job using Azure batch version 12.0**

**Notes: -**

**1-you have to note that Azure batch service 12.0 has big advantage then Azure batch service 9.0 that you can send the output result to container azure storage account.**

**using Microsoft.Azure.Batch;**

**using Microsoft.Azure.Batch.Auth;**

**using Microsoft.Azure.Batch.Common;**

**using Microsoft.WindowsAzure.Storage;**

**using Microsoft.WindowsAzure.Storage.Blob;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace BatchnewApp{**

**class Program{**

**// Add the batch account crdentials here**

**private const string demo\_batchAccountName = "enbehbatchaccount";**

**private const string demo\_batchAccountKey = "lGSLGM1XnPCtOQrDrnqGotc0sC4kiYXLv73u8xx0EaMdiSZj/Z3/CgWU7t6ZozjHvm2jp6kFjISPnQYGtLXdjg==";**

**private const string demo\_batchAccountUrl = "https://enbehbatchaccount.westeurope.batch.azure.com";**

**// Here add the storage account details**

**private const string demo\_storageAccountName = "enbehstorageaccount";**

**private const string demo\_storageAccountKey = "aBcpqRONCoYSb8bdfTLiIe3Q97vPvF9VgIIePI5J2z6pjkIVlUIWxj+HOi52o6cgXRiNVxVN4Hy7zF6l2iM1dw==";**

**// These are general values required for the batch service**

**private const string PoolId = "ffmpegpool";**

**private const string jobID = "video\_processor";**

**private const string demo\_packageid = "ffmpeg";**

**private const string demo\_packageversion = "3";**

**static void Main(string[] args){**

**try{CoreAsync().Wait();}**

**finally{**

**Console.WriteLine();**

**Console.WriteLine("Program complete");**

**Console.ReadLine();}}**

**private static async Task CoreAsync(){**

**BatchSharedKeyCredentials demo\_sharedKeyCredentials = new BatchSharedKeyCredentials(demo\_batchAccountUrl, demo\_batchAccountName, demo\_batchAccountKey);**

**using (BatchClient demo\_batchClient = BatchClient.Open(demo\_sharedKeyCredentials)){**

**// This method is used to create the pool**

**await PoolCreation(demo\_batchClient, PoolId);**

**// This method is used to create the job**

**await JobCreation(demo\_batchClient, jobID, PoolId);**

**// This method is used to create the task**

**await TaskCreation(demo\_batchClient, jobID);}}**

**private static async Task PoolCreation(BatchClient p\_batchClient, string p\_poolId){**

**Console.WriteLine("Creating the pool of virtual machines");**

**try{**

**var poolsLst = p\_batchClient.PoolOperations.ListPools().ToList();**

**var isExist = poolsLst.Where(x => x.Id == p\_poolId).FirstOrDefault();**

**if (isExist != null){return;}**

**//we define the image o.s**

**ImageReference demo\_image = new ImageReference(**

**publisher: "MicrosoftWindowsServer",**

**offer: "WindowsServer",**

**sku: "2016-Datacenter",**

**version: "latest");**

**VirtualMachineConfiguration demo\_configuration =**

**new VirtualMachineConfiguration(**

**imageReference: demo\_image,**

**nodeAgentSkuId: "batch.node.windows amd64");**

**CloudPool demo\_pool = null;**

**//it will create 1 dedicated pool of type Standard\_A1\_V2**

**demo\_pool = p\_batchClient.PoolOperations.CreatePool(**

**poolId: p\_poolId,**

**targetDedicatedComputeNodes: 1,**

**targetLowPriorityComputeNodes: 0,**

**virtualMachineSize: "STANDARD\_D1\_v2",**

**virtualMachineConfiguration: demo\_configuration);**

**//we assign the pool with package defined with id and version**

**demo\_pool.ApplicationPackageReferences = new List<ApplicationPackageReference>{**

**new ApplicationPackageReference{**

**ApplicationId = demo\_packageid,**

**Version = demo\_packageversion}};**

**//it will apply and create the pool**

**await demo\_pool.CommitAsync();}**

**catch (BatchException pool\_error) { Console.WriteLine(pool\_error.Message); }}**

**private static async Task JobCreation(BatchClient p\_batchClient, string p\_jobId, string p\_poolId){**

**//we get all jobs and check if the job exist or not to delete if exist and create new one**

**var lst = p\_batchClient.JobOperations.ListJobs().ToList();**

**var isExist = lst.Where(x => x.Id == p\_jobId).FirstOrDefault();**

**if (isExist != null) { p\_batchClient.JobOperations.DeleteJob(p\_jobId); }**

**Console.WriteLine("Creating the job");**

**//it will create job with id and assign pool to it**

**CloudJob demo\_job = p\_batchClient.JobOperations.CreateJob();**

**demo\_job.Id = p\_jobId;**

**demo\_job.PoolInformation = new PoolInformation { PoolId = p\_poolId };**

**//it will apply the configuration**

**await demo\_job.CommitAsync();}**

**private static async Task TaskCreation(BatchClient p\_batchClient, string p\_jobId){**

**//we define input and ouput containers**

**Console.WriteLine("Creating the Task");**

**string taskId = "demotask";**

**string in\_container\_name = "inputs";**

**string out\_container\_name = "outputs";**

**string l\_blobName = "1280.mp4";**

**string outputfile = "audio.aac";**

**string storageConnectionString = String.Format("DefaultEndpointsProtocol=https;AccountName={0};AccountKey={1}",**

**demo\_storageAccountName, demo\_storageAccountKey);**

**CloudStorageAccount l\_storageAccount = CloudStorageAccount.Parse(storageConnectionString);**

**//we create cloudblob client to access to blob containers**

**CloudBlobClient l\_blobClient = l\_storageAccount.CreateCloudBlobClient();**

**//we access to inputs container**

**CloudBlobContainer in\_container = l\_blobClient.GetContainerReference(in\_container\_name);**

**//we access to outputs container**

**CloudBlobContainer out\_container = l\_blobClient.GetContainerReference(out\_container\_name);**

**//we apply read , list permissions for inputs container**

**SharedAccessBlobPolicy i\_sasConstraints = new SharedAccessBlobPolicy{**

**SharedAccessExpiryTime = DateTime.UtcNow.AddHours(2),**

**Permissions = SharedAccessBlobPermissions.Read | SharedAccessBlobPermissions.List};**

**//we apply write permission for outputs container**

**SharedAccessBlobPolicy o\_sasConstraints = new SharedAccessBlobPolicy{**

**SharedAccessExpiryTime = DateTime.UtcNow.AddHours(2),**

**Permissions = SharedAccessBlobPermissions.Write};**

**//we generate SASS Token for input container and combine with the URL of the inputs container**

**string in\_sasToken = in\_container.GetSharedAccessSignature(i\_sasConstraints);**

**string in\_containerSasUrl = String.Format("{0}{1}", in\_container.Uri, in\_sasToken);**

**//we generate SASS Token for input container and combine with the URL of the outputs container**

**string out\_sasToken = out\_container.GetSharedAccessSignature(o\_sasConstraints);**

**string out\_containerSasUrl = String.Format("{0}{1}", out\_container.Uri, out\_sasToken);**

**//we declare resources file for the inputs container by set the input container url instead of write this command inputFile.Add(new ResourceFile(l\_blobSasUrl, l\_blobName));**

**ResourceFile inputFile = ResourceFile.FromStorageContainerUrl(in\_containerSasUrl);**

**List<ResourceFile> file = new List<ResourceFile>();**

**file.Add(inputFile);**

**string appPath = String.Format("%AZ\_BATCH\_APP\_PACKAGE\_{0}#{1}%", demo\_packageid, demo\_packageversion);**

**string taskCommandLine = String.Format("cmd /c {0}\\ffmpeg.exe -i {1} -vn -acodec copy audio.aac", appPath, l\_blobName);**

**CloudTask task = new CloudTask(taskId, taskCommandLine);**

**task.ResourceFiles = file;**

**// Setting the output file location and store the audio file inside the outputs container**

**//when the task success upload the result into the outputs container**

**List<OutputFile> outputFileList = new List<OutputFile>();**

**OutputFileBlobContainerDestination outputContainer = new OutputFileBlobContainerDestination(out\_containerSasUrl);**

**OutputFile outputFile = new OutputFile(outputfile,**

**new OutputFileDestination(outputContainer),**

**new OutputFileUploadOptions(OutputFileUploadCondition.TaskSuccess));**

**outputFileList.Add(outputFile);**

**task.OutputFiles = outputFileList;**

**await p\_batchClient.JobOperations.AddTaskAsync(p\_jobId, task);}}}**