

Microsoft

Exam 70-532

Developing Microsoft Azure Solutions

Version: 7.0

[Total Questions: 89]



Topic 1, Web-based Solution

Background

You are developing a web-based solution that students and teachers can use to collaborate on written assignments. Teachers can also use the solution to detect potential plagiarism, and they can manage assignments and data by using locally accessible network shares.

Business Requirements

The solution consists of three parts: a website where students work on assignments and where teachers view and grade assignments, the plagiarism detection service, and a connector service to manage data by using a network share.

The system availability agreement states that operating hours are weekdays between midnight on Sunday and midnight on Friday.

Plagiarism Service

The plagiarism detection portion of the solution compares a new work against a repository of existing works. The initial dataset contains a large database of existing works. Teachers upload additional works. In addition, the service itself searches for other works and adds those works to the repository.

Technical Requirements

Website

The website for the solution must run on an Azure web role.

Plagiarism Service

The plagiarism detection service runs on an Azure worker role. The computation uses a random number generator. Certain values can result in an infinite loop, so if a particular work item takes longer than one hour to process, other instances of the service must be able to process the work item. The Azure worker role must fully utilize all available CPU cores. Computation results are cached in local storage resources to reduce computation time.

Repository of Existing Works

The plagiarism detection service works by comparing student submissions against a repository of existing works by using a custom matching algorithm. The master copies of the works are stored in Azure blob storage. A daily process synchronizes files between blob storage and a file share on a virtual machine (VM). As part of this synchronization, the ExistingWorkRepository object adds the files to Azure Cache to improve the display performance of the website. If a student's submission is overdue, the Late property is set to the number of days that the work is overdue. Work files can be downloaded by using the Work action of the TeacherController object

Network Connector



Clients can interact with files that are stored on the VM by using a network share. The network permissions are configured in a startup task in the plagiarism detection service.

Service Monitoring

The CPU of the system on which the plagiarism detection service runs usually limits the plagiarism detection service. However, certain combinations of input can cause memory issues, which results in decreased performance. The average time for a given computation is 45 seconds. Unexpected results during computations might cause a memory dump. Memory dump files are stored in the Windows temporary folder on the VM that hosts the worker role.

Security

Only valid users of the solution must be able to view content that users submit. Privacy regulations require that all content that users submit must be retained only in Azure Storage. All documents that students upload must be signed by using a certificate named DocCert that is installed in both the worker role and the web role.

Solution Development

You use Microsoft Visual Studio 2013 and the Azure emulator to develop and test both the compute component and the storage component. New versions of the solution must undergo testing by using production data.

Scaling

During non-operating hours, the plagiarism detection service should not use more than 40 CPU cores. During operating hours, the plagiarism detection service should automatically scale when 500 work items are waiting to be processed. To facilitate maintenance of the system, no plagiarism detection work should occur during non-operating hours. All ASP.NET MVC actions must support files that are up to 2 GB in size.

Biographical Information

Biographical information about students and teachers is stored in a Microsoft Azure SQL database. All services run in the US West region. The plagiarism detection service runs on Extra Large instances.

Solution Structure

Relevant portions of the solution files are shown in the following code segments. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which the line belongs.

Diagnostics.wadcfg

```
DG01 <?xml version="1.0" encoding="utf-8" ?>
DG02 <DiagnosticMonitorConfiguration
DG03 xmlns="http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfiguration"
DG04 configurationChangePollInterval="PT1M"
DG05 overallQuotaInMB="4096">
CPERFORMANCECOUNTERS bufferQuotaInMB="0" scheduledTransferPeriod="PT30M">
CPERFORMANCECOUNTERS bufferQuotaInMB="0" scheduledTransferPeriod="\System\Context\subseteq bufferQuotaInMB="0" scheduledTransferPeriod
```



ExistingWorkRepository.cs

```
EW01 public static class ExistingWorkRepository
EW02 {
EW03
       public static void PopulateCache(string subject, string workId)
EW04
EW05
         var account = Storage.Account();
EW06
         var container = account.CreateCloudBlobClient().GetContainerReference("work" + subject);
EW07
         var body = container.GetBlockBlobReference(workId).DownloadText();
         var cache = new DataCacheFactory().GetCache(subject);
EW08
FW09
         cache.Add(workId, body);
EW11 }
```

PlagiarismCalculation.ps1

```
PC01 public class PlagiarismCalculation
PC02 {
PC03
       public double Compute(Work essay)
PC04
PC05
         var score = default(double);
PC06
         var account = Storage.Account();
PC07
         var cloudTableClient = account.CreateCloudTableClient();
         var cloudBlobClient = account.CreateCloudBlobClient();
PC08
         var existingWorks = cloudTableClient.GetTableReference("library").CreateQuery<Work>();
PC09
PC10
         var container = cloudBlobClient.GetContainerReference("work" + subject);
PC11
         foreach (var work in existingWorks.Execute())
PC12
           work.Body = container.GetBlockBlobReference(work.PartitionKey).DownloadText();
PC13
PC14
           score = GetMaxScore(essay, work, score);
PC15
PC16
         return score;
PC17
PC18
PC19
       private double GetMaxScore(Work work, Work previousWork, double previous)
PC20
        var rootPath = RoleEnvironment.GetLocalResource("ComputeResults").RootPath;
PC21
PC22
PC23
         return score;
PC24
PC25 }
```

SetupNetworkAccess.ps1

```
SN01 $acl = New-AzureAclConfig

SN02 Set-AzureAclConfig -AddRule -ACL $acl -Order 400 -Action permit`

-RemoteSubnet "192.168.5.1/24" -Description "Access for Northwood"

SN03 Set-AzureAclConfig -AddRule -ACL $acl -Order 200 -Action permit`

-RemoteSubnet "10.181.11.1/16" -Description "Access for Contoso, Ltd"

SN04 Get-AzureVM -ServiceName "FileService" -Name "FS" |`

Add-AzureEndpoint -Name "Files" -Protocol tcp -Localport 445`

-PublicPort 445 -ACL $acl | Update-AzureVM
```

TeacherController.cs

```
TC01 public class TeacherController : Controller
TC02 {
TC03
       public ActionResult Work(string workId, string subject)
TC04
TC05
TC06
TC07
      public ActionResult Upload(string workId, string subject)
TC08
TC09
TC10
TC11
      private static bool CheckDay(DateTime dt)
TC12
         if ((dt.DayOfWeek == DayOfWeek.Saturday) || (dt.DayOfWeek == DayOfWeek.Sunday))
TC13
TC14
           return true;
         return false;
TC15
TC16
      private static CloudQueueMessage BuildMessage(params string[] args)
TC17
TC18
TC19
         return new CloudQueueMessage(string.Join("/", args));
TC20
TC21 }
```

Work.cs

```
WK01 public class Work : TableEntity
WK02 {
       public string Body { get; set; }
WK03
WK04
       public string Author { get; set; }
WK05
       public bool IsReference { get; set; }
       public int Late { get; set; }
WK06
       [IgnoreProperty]
WK07
       public string Subject
WK08
WK09
WK10
         get { return RowKey; }
WK11
         set { RowKey = value; }
WK12
WK13
       [IgnoreProperty]
WK14
       public string WorkId
WK15
         get { return PartitionKey; }
WK16
         set { PartitionKey = value; }
WK17
WK18
WK19 }
```

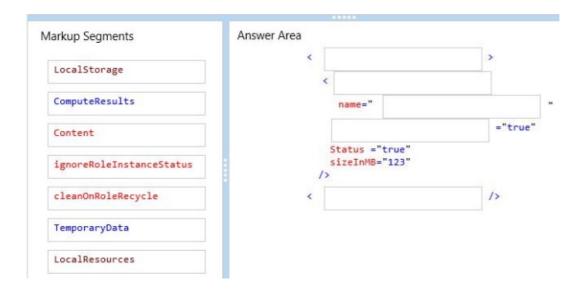
Microsoft 70-532: Practice Test

```
WorkerRole.cs
WR01 public class WorkerRole : RoleEntryPoint
WR02
WR03
       public override void Run()
WR04
         var account = Storage.Account();
WR05
WR06
         var queue = account.CreateCloudQueueClient().GetQueueReference("checkwork");
         var service = new PlagiarismCalculation();
WR07
WR08
         foreach (var queueMessage in GetWork(queue))
WR09
WR10
          var parts = queueMessage.AsString.Split(new[] {"/"},StringSplitOptions.None);
WR11
          service.Compute(parts[0], parts[1]);
WR12
WR13
WR14
       private IEnumerable<CloudQueueMessage> GetWork(CloudQueue queue)
WR15
WR16
WR17
WR18 }
```

Question No: 1 DRAG DROP - (Topic 1)

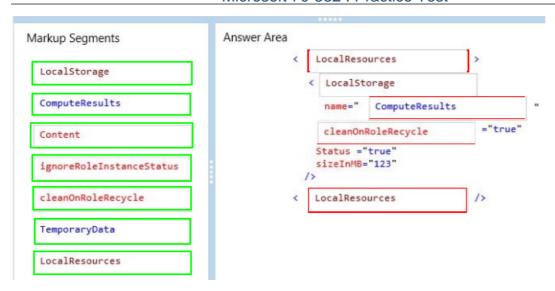
You need to configure storage for the solution.

What should you do? To answer, drag the appropriate XML segments to the correct locations. Each XML segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.



Answer:

Microsoft 70-532: Practice Test



Question No: 2 - (Topic 1)

You are deploying the web-based solution in the West Europe region.

You need to copy the repository of existing works that the plagiarism detection service uses. You must achieve this goal by using the least amount of time.

What should you do?

- **A.** Copy the files from the source file share to a local hard disk. Ship the hard disk to the West Europe data center by using the Azure Import/Export service.
- **B.** Create an Azure virtual network to connect to the West Europe region. Then use Robocopy to copy the files from the current region to the West Europe region.
- **C.** Provide access to the blobs by using the Microsoft Azure Content Delivery Network (CDN). Modify the plagiarism detection service so that the files from the repository are loaded from the CDN.
- **D.** Use the Asynchronous Blob Copy API to copy the blobs from the source storage account to a storage account in the West Europe region.

Answer: D

Explanation:

Ref: http://blogs.msdn.com/b/windowsazurestorage/archive/2012/06/12/introducing-asynchronous-cross-account-copy-blob.aspx

Question No: 3 HOTSPOT - (Topic 1)

You need to find all existing works about World History that are overdue and are stored in the repository.

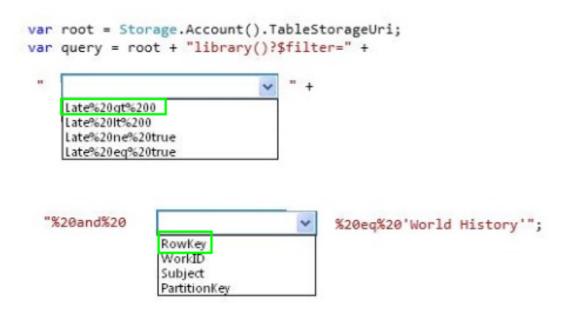
How should you complete the relevant code? To answer, select the appropriate option or options in the answer area.

Answer Area



Answer:

Answer Area

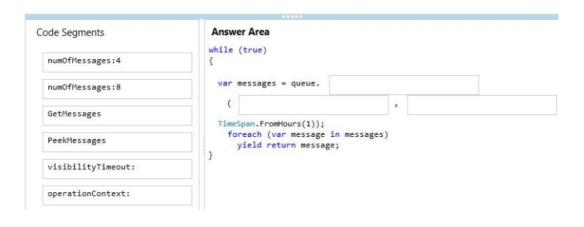




Question No: 4 DRAG DROP - (Topic 1)

You need to insert code at line WR16 to implement the GetWork method.

How should you complete the relevant code? To answer, drag the appropriate code segment to the correct location. Each code segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.



Answer:



Question No: 5 HOTSPOT - (Topic 1)

The Compute method in the PlagiarismCalculation class takes a significant amount of time to load existing works from blob storage. To improve performance, the service must load existing works from the cache.



You need to modify the Compute method in the class PlagiarismCalculation.

How should you modify the method? To answer, select the appropriate option or options in the answer area.

Answer Area

```
var existingWorks =

cloudTableClient.GetTableReference("library").CreateQuery<Work>();

var cache = new DataCache(essay.Author);
var cache = new DataCache(essay.Subject);
var cache = new DataCacheltemKey(essay.Author, "body");
var cache = new DataCacheltemKey(essay.Subject, "body");

foreach (var work in existingWorks.Execute())
{

    work.Body = cache.Get(work.Body).ToString();
    work.Body = cache.Get(work.RowKey).ToString();
    work.Body = cache.Get(work.Author).ToString();
    work.Body = cache.Get(work.PartitionKey).ToString();
    score = compute(essay, work, score);
}
```

Answer:

Answer Area

```
var existingWorks =
  cloudTableClient.GetTableReference("library").CreateQuery<Work>();
    var cache = new DataCache(essay.Author):
    var cache = new DataCache(essay.Subject);
    var cache = new DataCache(essay.Subject);
    var cache = new DataCache(essay.Subject, "body");

foreach (var work in existingWorks.Execute())
{
        work.Body = cache.Get(work.Body).ToString@;
        work.Body = cache.Get(work.RowKey).ToString@;
        work.Body = cache.Get(work.Author).ToString@;
        work.Body = cache.Get(work.PartitionKey).ToString@;
        work.Body = cache.Get(work.PartitionKey).ToString@;
        work.Body = cache.Get(work.PartitionKey).ToString@;
        work.Body = cache.Get(work.PartitionKey).ToString@;
        work.Body = cache.Get(work.PartitionKey).ToString@;
}
```