MapReduce on Windows Azure

Daytona – Client Application User Guide

Table of Contents

1 Overview

This document details out the usage of MapReduce client tool (mrclient.exe). The tool is a simple console application, created using Daytona framework APIs, to perform common operations like application submission, monitoring etc. against a deployment from command-line. The tool also serves a purpose to showcase the steps to build applications using Daytona framework.

2 Technical Details

The tool has a dependency on the following assemblies and these should be present in the same directory as the tool:

- Microsoft.WindowsAzure.StorageClient.dll
- Research.MapReduce.Core.dll

2.1 Syntax

```
mrclient [-cs <connection string>] -c <command> [<switch>...] [-h] [-?]
```

By default, the connection string specified by 'StorageConnectionString' setting in the application configuration file is used to connect to the Windows Azure storage account. This connection string can be overridden by the optional '–cs' argument on command-line.

The command to be executed (e.g. to upload a package) is specified by the mandatory '–c' argument. Details of these commands are provided in sections below. A command may require a set of command-specific switches to work. If any, these are provided as command-line arguments following the command name.

If the optional '–h' or '-?' arguments are specified while invoking the tool, all other arguments are ignored and the tool just prints a usage string.

Either a hyphen character (-) or a front-slash (/) can be used as the argument specifier.

In the event of a parser or command error, the response starts with 'Error:' as shown below:

```
D:\>mrclient.exe
Error: '-c' argument must be provided exactly once.
…
Use '-?' for help.
```

If a command executes successfully, the command-specific response is returned as shown below:

```
D:\>mrclient.exe -c createpkg -name wc -dir
"D:\MSR_CCF_Engagements\Daytona\Source\Research.MapReduce.Samples.WordCount\b
in\Debug" -searchpat "*.dll|*.exe"
```

```
PackageOwner: v-samaur
PackageName: wc
PackageDirectory:
D:\MSR_CCF_Engagements\Daytona\Source\Research.MapReduce.Samples.WordCount\bi
n\Debug
SearchPatterns: *.dll|*.exe
```

3 Commands

3.1 createpkg

This command creates a new application package in the Windows Azure storage account used by Daytona.

Switch	Required	Description
owner	False	Name of the package owner. If not specified, current logged in user is considered as the package owner.
name	False	Name of the created package. If not specified, a name will be auto-generated.
dir	False	Full path to the directory where assemblies of the package are located. If not specified, the current working directory is treated as the package directory.
searchpat	False	Search pattern to identify the files to be included in the package from the package directory. By default, a search pattern of *.dll is used which means that all files with .dll extension are treated as package files. Multiple search patterns can be used by separating them with a pipe (|) character. For example, to search for both *.dll and *.exe patterns, use a combined pattern of *.dll|*.exe.

A sample usage looks like:

```
mrclient.exe -c createpkg -name wc -dir
"D:\MSR_CCF_Engagements\Daytona\Source\Research.MapReduce.Samples.WordCount\b
in\Debug" -searchpat "*.dll|*.exe"
```

The command response has the following format:

```
PackageOwner: <Package owner>
PackageName: <Created package name>
PackageDirectory: <Package directory used>
SearchPatterns: <Search patterns used>
```

3.2 deletepkg

This command deletes an application package from the storage.

Switch	Required	Description
owner	False	Name of the package owner. If not specified, current logged in user is

		considered as the package owner.
name	True	Name of the package to be deleted.

A sample usage looks like:

```
mrclient.exe -c deletepkg -name wc
```

The command returns an empty response.

3.3 submitapp

This command submits a MapReduce application for execution. Switches other than those specified below are considered as controller arguments.

Switch	Required	Description
owner	False	Name of the application owner. If not specified, current logged in user is considered as the application owner.
name	False	The application name. If not specified, a name is generated by the system.
ctr	True	Assembly qualified name of the controller's type.
reusepkgowner	False	If an existing package is to be used, this switch defines the package owner. If 'reusepkgname' is specified and this switch is not specified, current logged in user is considered as the package owner.
reusepkgname	False	Name of an existing package to be used in the application. This switch is mutually exclusive to 'newpkgowner', 'newpkgname, `dir' and `searchpat' switches.
newpkgowner	False	Owner of the created package if `reusepkgname' switch is not specified. If this switch is also not specified, current logged in user name is considered.
newpkgname	False	Name of the created package if 'reusepkgname' switch is not specified. If this switch is also not specified, the package name is auto generated.
dir	False	Directory hosting the package assemblies if 'reusepkgname' switch is not specified. If this switch is also not specified, the current working directory is treated as the package directory.
searchpat	False	Search pattern used to search the package directory if 'reusepkgname' switch is not specified. If this switch is also not specified, a search pattern of *.dll is used.

A sample usage looks like:

```
mrclient -c submitapp -name "word count 1" -newpkgname wcpkg -dir
D:\MSR_CCF_Engagements\Daytona\Test\bin\Debug -ctr
"Research.MapReduce.Test.Common.WCLib.Controller, Research.MapReduce.Test,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"
```

The command responds with the following format.

```
ApplicationOwner: <Application owner>
ApplicationName: <Application name>
ReusedPackageOwner: <Existing package owner >
ReusedPackageName: <Name of the existing package>
NewPackageOwner: <Owner of the package if created>
NewPackageName: <Name of the package if created>
PackageDirectory: <Package directory if a package is created>
SearchPatterns: <Search patterns used if a package is created>
```

3.4 listapps

This command lists applications submitted to the service which pass the filtering criteria specified by the switches.

Switch	Required	Description
owner	False	List applications owned by the specified user. If not specified, all applications irrespective of the owner are listed.
createdafter	False	List applications that were created after this datetime. If not specified, a datetime of 1900-01-01 is considered.
createdbefore	False	List applications that were created before this datetime.If not specified, current UTC datetime is considered.
state	False	List applications that have this state. If not specified, all applications irrespective of the state are listed.

A sample usage looks like:

```
mrclient -c listapps -createdafter 2011-1-1
```

The command responds with the following format.

```
Owner: <Owner filter>
CreatedAfter: <CreatedAfter filter>
CreatedBefore: <CreatedBefore filter>
State: <State filter>
Applications:
User    Name    DateCreated     DateStarted     DateCompleted   State
----    ----    -----------     -----------     -------------   -----
<User>  <App>   <DateCreated>   <DateStarted>   <DateCompleted> <State>
...
```

3.5 getapp

This command gets the details of an application.

Switch	Required	Description
owner	False	Name of the application owner. If not specified, current logged in user is considered as the application owner.
name	True	The application name.

logdir	False	If specified, an application execution report, named *App_<AppOwner>_<AppName>.csv*, is created in this directory. While preparing the report, this directory is searched for any existing logs for the application. If found, the execution report is prepared from these logs. Otherwise, application logs are downloaded from the table storage and then the report is prepared.
forcedownload	False	This switch can be used only in combination with the 'logdir' switch. If set to 'True', application logs are downloaded from the table storage even if they are present in the log directory. This is useful, for example, when the application is still running and its latest execution information is to be retrieved. Or, only a subset of application logs were downloaded earlier due to an incorrect time range provided to the 'getlogs' command.

A sample usage looks like:

```
mrclient -c getapp -name "word count 1"
```

The command responds with the following format.

```
ApplicationOwner: <Owner of the application>
LogDirectory: <Directory hosting the log information>
ForceDownload: <Forcedownload switch>
DateCreated: <DateCreated of the application>
DateStarted: <DateStarted of the application>
DateCompleted: <DateCompleted of the application>
State: <State of the application>
FailReason: <FailReason of the application>
Arguments:
        <ArgumentName>: <ArgumentValue>
        ...
Results:
        <ResultName>: <ResultValue>
        ...
```

If the 'logdir' switch is specified, two report files *App_<AppOwner>_<AppName>.csv* and *App_<AppOwner>_<AppName>_MapOutputDownloadAttempts.csv* are created in the log directory. The former contains various metrics about the application, its jobs and tasks while the latter contains details about the attempts made by reducers to download the map phase output.

The *App_<AppOwner>_<AppName>.csv* file contains comma-separated values with following columns:

Column Name	Description
AppOwner	Owner of the application.
AppName	Name of the application.

AppId	The internal unique identifier assigned to the application.
AppState	Can be one of {Waiting, Running, Concluded}
AppWaitDurationInMs	Duration of the period when the application was in a waiting loop due to a limit on concurrent applications.
AppStartTime	Time when the application started.
AppEndTime	Time when the application concluded.
AppExecDurationInMs	Execution duration of the application.
AppError	Error encountered during application execution.
CtrCreateTime	Time when the controller was created.
CtrCreateDurationInMs	Time taken to download the package and instantiate the controller.
CtrStartTime	Time when the controller started.
CtrEndTime	Time when the controller concluded.
CtrExecDurationInMs	Execution duration of the controller.
CtrResultSaveDurationInMs	Time taken to save Controller results.
JobId	An identifier generated for the job instance when it was created. The same job instance can be executed multiple times, so this value does not uniquely identify a job execution. To uniquely identify the execution, use this value in conjunction with *JobIterationCount* below.
JobIterationCount	Identifies the job iteration when the same job instance is executed multiple times.
JobExecutionId	The unique identifier assigned to each job iteration.
JobState	Can be one of {Started, MapPhase, ReducePhase, Concluded}.
JobStartTime	Time when Job.Run() is invoked.
JobEndTime	Time when the job concluded.
JobExecDurationInMs	Execution duration of the job.
JobError	Error encountered during job execution.
JobInitDurationInMs	Time taken to invoke the GetPartitions method of the IDataPartitioner and initialize other structures for the Job.
TaskType	Can be one of {Map, Reduce}
TaskNumber	Index of this task.
TaskDispatchCount	When a slave crashes, tasks running on that slave are re-dispatched to other active slaves. This number identifies the dispatch count of this task.
TaskDispatchId	The unique identifier assigned to a task

	dispatch.
TaskTrackerId	Identifies the task tracker that executed this task.
RoleInstance	Identifies the Windows Azure role instance (VM) that execution this task.
MapState	Can be one of {Dispatched, Created, Running, Concluded}
MapDispatchTime	Time when the task was dispatched to a slave for execution.
MapCreateTime	Time when the task is constructed on a slave.
MapCreateDurationInMs	The duration to download Application package and initialize other task related structures.
MapCreateError	Error encountered during task creation.
MapTaskStartTime	Time when the task started its execution.
MapTaskEndTime	Time when the task concluded its execution.
MapTaskExecDurationInMs	Duration of task execution.
MapTaskExecError	Error encountered during task execution.
MapperConfigureDurationInMs	Time taken to complete IMapper.Configure invocation.
GetRecordsDurationInMs	Time taken to complete IRecordReader.GetRecords invocation.
MapInputEnumerationDurationInMs	Cumulative time taken to enumerate all input records.
MapInputRecordCount	Total number of input records.
MapperExecDurationInMs	Cumulative time taken to complete all IMapper.Map invocations.
KeyPartConfigureDurationInMs	Time taken to complete IKeyPartitioner.Configure invocation.
MapOutputSaveDurationInMs	Cumulative time taken to complete all MapOutputCollection.Add invocations.
MapOutputRecordCount	Total number of output records produced by the Map() invocations.
CombinerOutputRecordCount	Total number of output records produced by the combiners.
MapResponseTime	Time when MapResponse is sent from the slave.
ReduceState	Can be one of {InitDispatched, Created, StartDispatched, Ready, Started, Downloading, Running, Concluded}
ReduceInitDispatchTime	Time when the InitilizeReduceRequest is sent to a slave.
ReduceCreateTime	Time when the task is constructed on a slave.
ReduceCreateDurationInMs	The duration to download Application package and initialize other task related structures.

ReduceCreateError	Error encountered during task creation.
ReduceStartDispatchTime	Time when the StartReduceRequest is sent to the slave.
ReduceReadyTime	Time when the Reduce task is queued for execution. This happens in response to receiving the StartReduceRequest by the slave.
ReduceTaskStartTime	Time when the task started its execution.
ReduceDownloadConcludeTime	Reduce task waits for the Map phase output to be downloaded before invoking IReducer etc. This value captures the time when the Reduce task comes out of the wait loop.
ReduceDownloadWaitDurationInMs	Duration when the task was waiting for the map output to be downloaded.
ReduceTaskEndTime	Time when the task concluded its execution.
ReduceTaskExecDurationInMs	Duration of task execution.
ReduceTaskExecError	Error encountered during task execution.
ReducerConfigureDurationInMs	Time taken to complete the invocation of IReducer.Configure.
ReduceInputRecordCount	Total number of input records.
ReduceInputEnumerationDurationInMs	Cumulative time taken to enumerate all input records.
ReducerExecDurationInMs	Cumulative time taken to complete all IReducer.Reduce invocations.
ReduceInvocations	Total number of Reduce() invocations. This is generally equal to the number of distinct keys present in the Reducer input.
RecordWriterWriteDurationInMs	Total time taken by IRecordWriter.Write() to complete. The method in turn enumerates input records, invokes IReducer.Reduce, persists output and executes other custom logic.
ReduceOutputRecordCount	Total number of output records. This is generally equal to the number of Reduce() invocations.
ReduceResponseTime	Time when the ReduceResponse is sent from the slave.

The report file contains information in a de-normalized fashion with many empty columns in a row. For example, the row containing summary information of an application has empty Job and Task related columns. Likewise, the summary row of a Job has empty Task columns and all application-level metric columns are also empty as shown below.

AppOwner	AppName	JobId	JobIteration	AppState	...	JobState	JobStartTime
alice	w-count			Running			

alice	w-count	3A1E2…	1			MapPhase	2011-01-01T…

The *App_<AppOwner>_<AppName>_MapOutputDownloadAttempts.csv* file contains comma-separated values with following columns:

Column Name	Description
TaskDispatchId	Dispatch id of the Reduce Task.
MapTask	Map task number.
AttemptId	The unique identifier assigned to this attempt.
StartTime	Download start time.
ConcludeTime	Download complete time.
DurationInMs	Download duration.
RecordCount	Count of the records downloaded.
Retries	The number of times this download was retried.
Error	Error encountered during download.

3.6 abortapp

This command requests the service to abort a running application.

Switch	Required	Description
owner	False	Name of the application owner. If not specified, current logged in user is considered as the application owner.
name	True	The application name.

A sample usage looks like:

```
mrclient -c abortapp -name "word count 1"
```

The command returns an empty response.

3.7 deleteapp

This command deletes application information from the storage.

Switch	Required	Description
owner	False	Name of the application owner. If not specified, current logged in user is considered as the application owner.
name	True	The application name.

A sample usage looks like:

```
mrclient -c deleteapp -name "word count 1"
```

The command returns an empty response.

3.8 getlogs

This command downloads Daytona logs generated within a specified time range. The command also formats the downloaded logs in a more readable format to assess the state and performance of applications that executed within the time range.

Switch	Required	Description
start	True	Log entries created after this time will be downloaded.
end	True	Log entries created before this time will be downloaded.
dir	False	Directory where logs will be download. If not specified, current working directory is treated as the log directory.
threads	False	Number of threads to employ while downloading the logs.

A sample usage looks like:

```
mrclient -c getlogs -start 2011-11-09T00:00:00Z -end 2011-11-09T00:10:00Z
```

The command responds with the following format.

```
LogDirectory: <The log directory>
Threads: <Threads employed>
```

After execution, the command creates a sub-directory named <start>_<end> in the log directory that hosts the log information for the requested time range. If the sub-directory already exists, it is overwritten. Multiple files containing data as comma-separated values are then created in this sub-directory. Sub-sections below describe the content of these files.

3.8.1 Raw logs

A file named *WADLogsTable.csv* is created that contains all log entries as they are present in the WADLogsTable within the requested time range. The file contains comma-separated values with following columns:

Column Name	Description
PartitionKey	Partition key of this entry.
RowKey	Row key of this entry.
Timestamp	Time when this entry was created.
EventTickCount	Time when the event responsible for creating this entry was raised.
DeploymentId	Id of the Daytona cluster deployment.
Role	E.g. Research.MapReduce.CloudHost.Slave.
RoleInstance	Identifies a VM.
Level	Event level.
EventId	Event type.
Pid	Process that raised this event.
Tid	Thread that raised this event.
Message	Event message.

3.8.2 Applications

A file named *Applications.csv* is created that contains a list of all applications that began their lifecycle within the requested time range. The fact that the application has started is inferred from log entries - specifically an entry corresponding to the 'Application Downloaded' event. The file contains comma-separated values with following columns:

Column Name	Description
ApplicationOwner	Owner of the application.
ApplicationName	Name of the application.
ApplicationId	An internal identifier generated by the framework to uniquely identify the application.
AppState	Can be one of {Waiting, Running, Concluded}
AppWaitDurationInMs	Duration of the period when the application was in a waiting loop due to a limit on concurrent applications.
AppStartTime	Time when the application started.
AppEndTime	Time when the application concluded.
AppExecDurationInMs	Execution duration of the application.
AppError	Error encountered during application execution.
CtrCreateTime	Time when the controller was created.
CtrCreateDurationInMs	Time taken to download the package and instantiate the controller.
CtrStartTime	Time when the controller started.
CtrEndTime	Time when the controller concluded.
CtrExecDurationInMs	Execution duration of the controller.
CtrResultSaveDurationInMs	Time taken to save Controller results.

Since the information above is inferred from log entries within the requested time range, some of the columns may be empty if log entries are missing. For example, if there is no log entry corresponding to an 'Application Concluded' event, 'AppEndTime' and 'AppExecDurationInMs' will not be populated. This might happen if the time range is not properly selected or the row corresponding to the 'Application Concluded' event is yet to be shipped to WADLogsTable.

3.8.3 Jobs

A file named *Jobs.csv* is created that contains state and performance related information of jobs as inferred from log entries. Each row represents an execution of a job instance. A job instance, during its lifetime, can undergo execution multiple times. The file contains comma-separated values with following columns:

Column Name	Description
AppId	Unique identifier of the parent application as described above.
JobId	An identifier generated for the job instance when it was created. The same job instance can be executed multiple times, so this value does not uniquely identify a job execution. To uniquely identify the execution, either use this value in conjunction with *JobIterationCount* below or use the surrogate key *JobExecutionId*.
JobExecutionId	Each job execution is uniquely identified by this value.
JobIterationCount	Identifies the job iteration when the same job instance is executed multiple times.
JobState	Can be one of {Started, MapPhase, ReducePhase, Concluded}.
JobStartTime	Time when Job.Run() is invoked.
JobEndTime	Time when the job concluded.
JobExecDurationInMs	Execution duration of the job.
JobError	Error encountered during job execution.
JobInitDurationInMs	Time taken to invoke the GetPartitions method of the IDataPartitioner and initialize other structures for the Job.

3.8.4 Map Tasks

A file named *MapTasks.csv* is created that contains state and performance related information of Map tasks as inferred from log entries. Each row represents one dispatch of a Map task. The file contains comma-separated values with following columns:

Column Name	Description
JobExecutionId	Execution based unique identifier of the parent job as described above.
TaskNo	Index of this task.
TaskDispatchCount	When a slave crashes, tasks running on that slave are re-dispatched to other active slaves. This number identifies the dispatch count of this task.
TaskDispatchId	Each task dispatch is allotted this system generated unique identifier.
TaskTrackerId	Identifies the task tracker that executed this task.
RoleInstance	Identifies the Windows Azure role instance (VM) that executed this task.
MapState	Can be one of {Dispatched, Created, Running, Concluded}
MapDispatchTime	Time when the task was dispatched to a slave for execution.
MapCreateTime	Time when the task is constructed on a slave.
MapCreateDurationInMs	The duration to download Application package and initialize other task related structures.
MapCreateError	Error encountered during task creation.

MapTaskStartTime	Time when the task started its execution.
MapTaskEndTime	Time when the task concluded its execution.
MapTaskExecDurationInMs	Duration of task execution.
MapTaskExecError	Error encountered during task execution.
MapperConfigureDurationInMs	Time taken to complete IMapper.Configure invocation.
GetRecordsDurationInMs	Time taken to complete IRecordReader.GetRecords invocation.
MapInputEnumerationDurationInMs	Cumulative time taken to enumerate all input records.
MapInputRecordCount	Total number of input records.
MapperExecDurationInMs	Cumulative time taken to complete all IMapper.Map invocations.
KeyPartConfigureDurationInMs	Time taken to complete IKeyPartitioner.Configure invocation.
MapOutputSaveDurationInMs	Cumulative time taken to complete all MapOutputCollection.Add invocations.
MapOutputRecordCount	Total number of output records produced by the Map() invocations.
CombinerOutputRecordCount	Total number of output records produced by the combiners.
MapResponseTime	Time when MapResponse is sent from the slave.

3.8.5 Reduce Tasks

A file named *ReduceTasks.csv* is created that contains state and performance related information of Reduce tasks as inferred from log entries. Each row represents one dispatch of a Reduce task. The file contains comma-separated values with following columns:

Column Name	Description
JobExecutionId	Execution based identifier of the parent job as described above.
TaskNo	Index of this task.
TaskDispatchCount	When a slave crashes, tasks running on that machine are re-dispatched to other active slaves. This number identifies the dispatch count of this task.
TaskDispatchId	Each task dispatch is allotted this system generated unique identifier.
TaskTrackerId	Identifies the task tracker that executed this task.
RoleInstance	Identifies the Windows Azure role instance (VM) that executed this task.
ReduceState	Can be one of {InitDispatched, Created, StartDispatched, Ready, Started, Downloading, Running, Concluded}
ReduceInitDispatchTime	Time when the InitilizeReduceRequest is sent to a

	slave.
ReduceCreateTime	Time when the task is constructed on a slave.
ReduceCreateDurationInMs	The duration to download Application package and initialize other task related structures.
ReduceCreateError	Error encountered during task creation.
ReduceStartDispatchTime	Time when the StartReduceRequest is sent to the slave.
ReduceReadyTime	Time when the Reduce task is queued for execution. This happens in response to receiving the StartReduceRequest by the slave.
ReduceTaskStartTime	Time when the task started its execution.
ReduceDownloadConcludeTime	Reduce task waits for the Map phase output to be downloaded before invoking IReducer etc. This value captures the time when the Reduce task comes out of the wait loop.
ReduceDownloadWaitDurationInMs	Duration when the task was waiting for the map output to be downloaded.
ReduceTaskEndTime	Time when the task concluded its execution.
ReduceTaskExecDurationInMs	Duration of task execution.
ReduceTaskExecError	Error encountered during task execution.
ReducerConfigureDurationInMs	Time taken to complete the invocation of IReducer.Configure.
ReduceInputRecordCount	Total number of input records.
ReduceInputEnumerationDurationInMs	Cumulative time taken to enumerate all input records.
ReducerExecDurationInMs	Cumulative time taken to complete all IReducer.Reduce invocations.
ReduceInvocations	Total number of Reduce() invocations.
RecordWriterWriteDurationInMs	Total time taken by IRecordWriter.Write() to complete. The method in turn enumerates input records, invokes IReducer.Reduce, persists output and executes other custom logic.
ReduceOutputRecordCount	Total number of output records.

3.8.6 Map Output Download Attempts

A file named *MapOutputDownloadAttempts.csv* is created that contains the details of each attempt from reducers to download map phase data. The file contains comma-separated values with following columns:

Column Name	Description
TaskDispatchId	Dispatch id of the Reduce Task.
MapTask	Map task number.
AttemptId	The unique identifier assigned to this attempt.

StartTime	Download start time.
ConcludeTime	Download complete time.
DurationInMs	Download duration.
RecordCount	Count of the records downloaded.
Retries	The number of times this download was retried.
Error	Error encountered during download.

3.9 getcounters

This command downloads Daytona performance counters and translates them into a more readable format so that they can be correlated with application execution info retrieved using other commands (e.g. getlogs and getapp).

Switch	Required	Description
start	True	Log entries created after this time will be downloaded.
end	True	Log entries created before this time will be downloaded.
dir	False	Directory where logs will be download. If not specified, current working directory is treated as the log directory.

The command responds with the following format.

```
LogDirectory: <Directory to download counters to>
```

After execution, the command creates a sub-directory named <start>_<end> in the log directory that hosts the performance counters information for the requested time range. If the sub-directory already exists, it is overwritten. Multiple files containing data as comma-separated values are then created in this sub-directory. Each file corresponds to a performance counter and has the following columns:

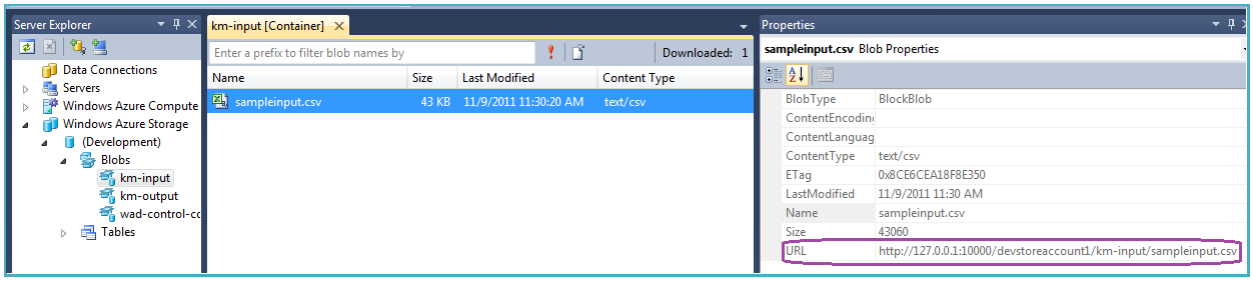
Column Name	Description
RoleInstance	Identifies the Windows Azure role instance that generated this counter value.
EventTickCount	Time at which the counter value was recorded.
Timestamp	Time at which this row was created. This may differ from when the counter was actually recorded.
Value	Value of the counter.

4 Sample – K-Means Clustering

This section exhibits the usage of client tool to submit and monitor K-means clustering example.

4.1 Upload Input Data

Using any of the available tools to access Windows Azure Storage (e.g. Azure Storage Explorer), upload the *Research.MapReduce.Samples.KMeansClustering\Resources\sampleinput.csv* file to a blob container say '*km-input*'. Figure below shows the uploaded blob using Visual Studio server explorer. Take note of the input blob URL which is used later as a controller argument. Also, create a container to contain the output data say '*km-output*'.



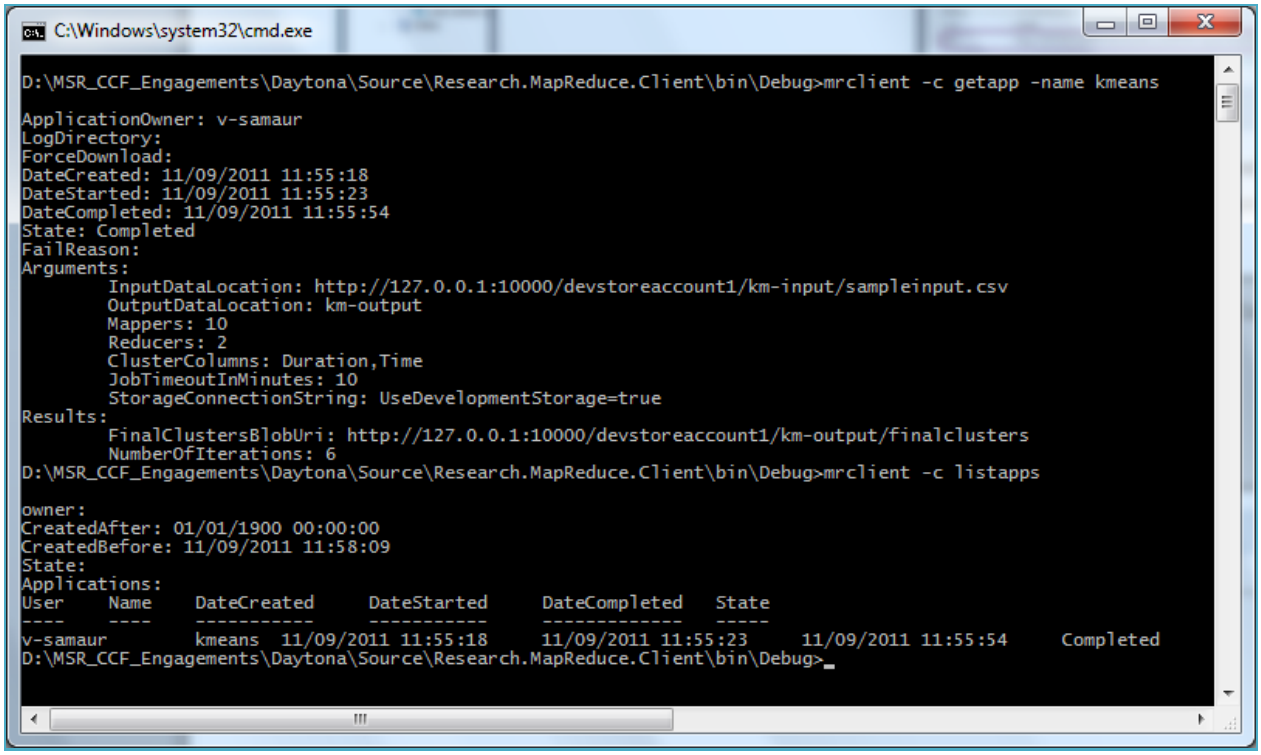
4.2 Submit Application

To submit an application from client tool, use 'submitapp' command. The same command can be used to create a new package for the application and set controller's parameters. A sample command is shown below:

```
mrclient -c submitapp -name "kmeans" -newpkgname km-pkg -dir
"D:\MSR_CCF_Engagements\Daytona\Source\Research.MapReduce.Samples.KMeansClust
ering\bin\Debug" -searchpat "*.dll|*.exe" -ctr
"Research.MapReduce.Samples.KMeansClustering.KMeansClusteringController,
Research.MapReduce.Samples.KMeansClustering, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=null" -InputDataLocation
http://127.0.0.1:10000/devstoreaccount1/km-input/sampleinput.csv -
OutputDataLocation km-output -Mappers 10 -Reducers 2 -ClusterColumns
Duration,Time -JobTimeoutInMinutes 10
```

4.3 Monitor Application

Use the 'getapp' or 'listapps' command to check the status of this application. Figure below shows a sample output.



After the logs have been shipped to Windows Azure storage, use the 'getapp' command with '–logdir' switch to generate various execution reports.

```
mrclient -c getapp -name kmeans -logdir d:\applogs -forcedownload true
```