If you want your Azure Cloud Services or Azure Web App application to use two or more instances (processes dedicated to a Web App or web roles in a Cloud Service), you must configure where the application will store session state information. Session state must be stored either in the database, which is shared among the servers, or in a special service. Session state cannot be stored in the server's memory, because the session would be lost if the web farm switched requests to another instance.

Options for storing session state information in the Azure environment:

- In Microsoft Azure SQL Database – easy to set up, suitable for small projects or projects with read access to web pages.
- In Microsoft Azure Redis Cache – a dedicated cache service appropriate for more demanding projects. See the Azure Redis Cache Documentation.

Storing session state information in Azure SQL Database

1. Open your Azure project in Visual Studio.
2. Right-click the **CMSApp** project and select **Manage NuGet packages**.
3. Install the **Microsoft.AspNet.Providers** package.
4. Open the **web.config** file.
5. Follow the instructions in the code comments of the **sessionState** section.

After this, your project is configured to store session state information in the Microsoft Azure SQL Database.

Storing session state information in Azure Redis Cache

To configure your project to use Azure Redis Cache:

1. Create a new Redis Cache according to the instructions in the Use Azure Redis Cache with a .NET application article.
2. Open the project in Visual Studio.
3. Right-click the solution and select **Manage NuGet Packages**.
4. Search for and install the **Microsoft.Web.RedisSessionStateProvider** package.
    - This action will automatically add all required assembly references to the project and create the following section in the web.config file:

```
<sessionState mode="Custom" customProvider="MySessionStateStore">
  <providers>
    <!--
      <add name="MySessionStateStore"
        host = "127.0.0.1" [String]
        port = "" [number]
        accessKey = "" [String]
        ssl = "false" [true|false]
        throwOnError = "true" [true|false]
        retryTimeoutInMilliseconds = "0" [number]
        databaseId = "0" [number]
        applicationName = "" [String]
        connectionTimeoutInMilliseconds = "5000" [number]
        operationTimeoutInMilliseconds = "5000" [number]
      />
    -->
    <add name="MySessionStateStore" type="Microsoft.Web.Redis.
RedisSessionStateProvider" host="mycache.redis.cache.windows.net"
    accessKey="..." ssl="true" />
  </providers>
</sessionState>
```

    - Specify the **host** name and **accessKey** values which you can find in the properties of the cache service on Azure.

Instead of inserting the values of the *host* and *accessKey* attributes directly, you can specify them as key-value pairs in the *appSettings* section of the application's web.config file. You can then reference the values anywhere in the configuration file using their assigned keys.

```
<add key="MyRedisAccessKey" value="...">
```

Values stored in the *appSettings* section can be easily accessed and managed, for example, via the Azure Portal. The contents of the section can also be encrypted, preventing untrustworthy people from accessing application secrets, such as the Redis cache provider access key.

See the official AspNet Redis documentation for details.

5. Comment out the default session state provider in the web.config file.
   - By default, it is the *InProc* provider:

```
<!-- <sessionState mode="InProc" customProvider="DefaultSessionProvider">
  <providers>
    <add name="DefaultSessionProvider" type="System.Web.Providers.
DefaultSessionStateProvider, System.Web.Providers, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" connectionStringName="
DefaultConnection" />
  </providers>
</sessionState> -->
```

You can now deploy your project to Azure and start utilizing Azure Redis Cache as a session state provider.