# Demo script

## Downloading and installing Service Bus Explorer

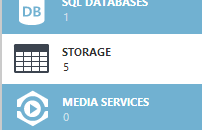
Navigate to <http://code.msdn.microsoft.com/windowsazure/Service-Bus-Explorer-f2abca5a> and unpack. Navigate to the C#/bin directory and copy onto the desktop. We’ll be using Service Bus Explorer later to demo.

## Install Azure Management Studio

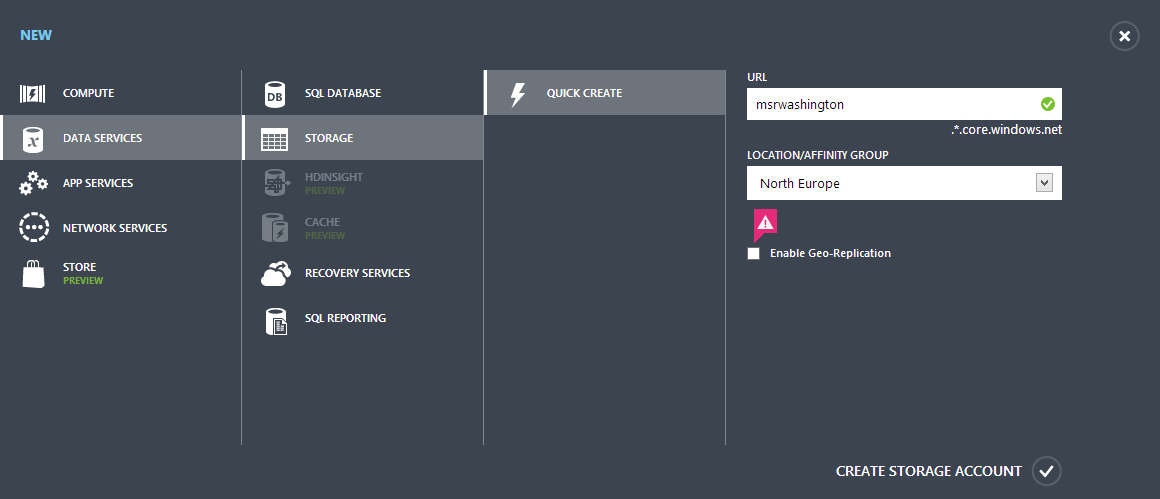
Obtain a trial version or license from cerebrata.com. Thirty day trials are available. Place a link on the start menu or desktop for easy access.

## Create storage account

Create a storage account through the Windows Azure portal. Navigate to the storage tab on the left



At the bottom left of the screen press **+NEW** button. Enter the name and region as per the diagram below.



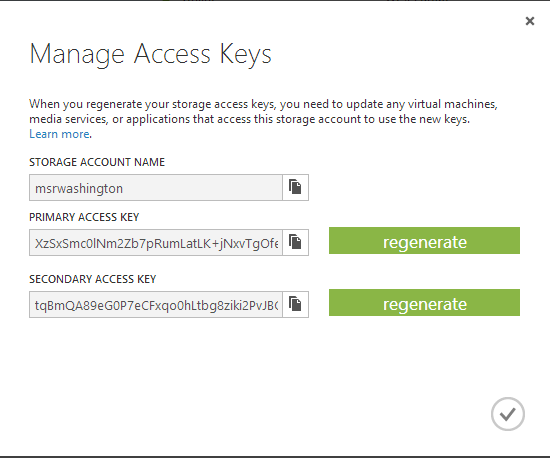
You should see it creating like so in the results pane.



Click on the **manage access keys** button at the bottom of the screen.



Copy the primary access key in the resulting dialogue.



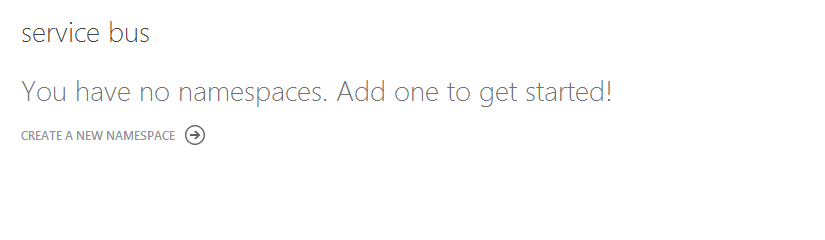
We’ll be updating the configuration code in the software to reflect the new storage account in a section later on.

## Create service bus namespace

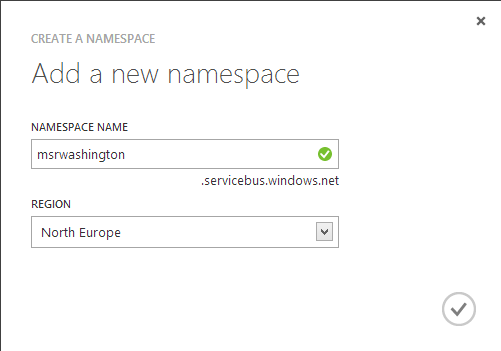
Navigate to the service bus tab in the Windows Azure management portal.



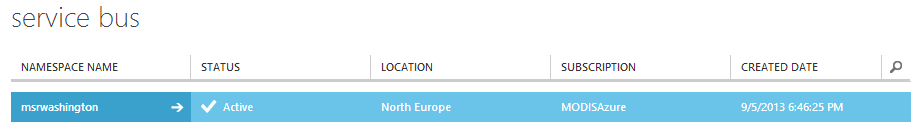
On the results pane click on **create a new namespace**.



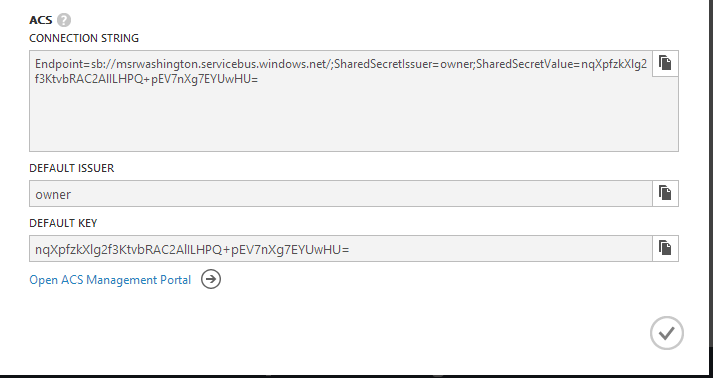
Enter the details in the dialogue as per below.



The service bus should have been created like so as per the results pane:



The service bus connection information button should be visible at the bottom of the screen. Copy the connection string, default issuer and default key. We’ll be using them later in our client-server application.

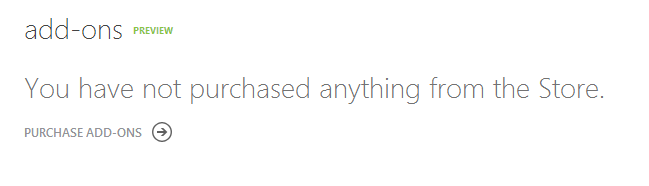


## Create an instance of Mongolabs

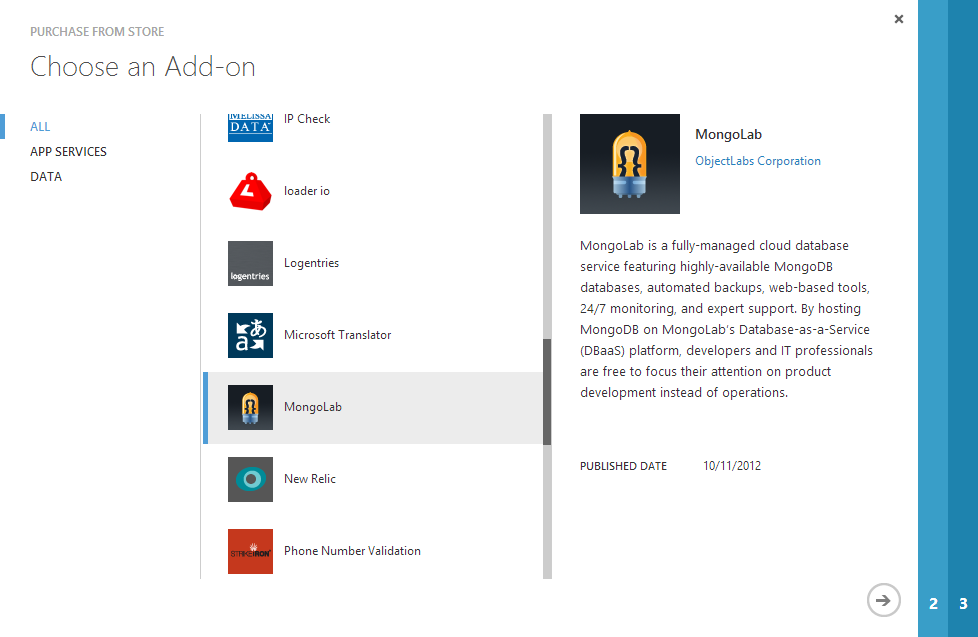
Navigate to the add ons dialogue:



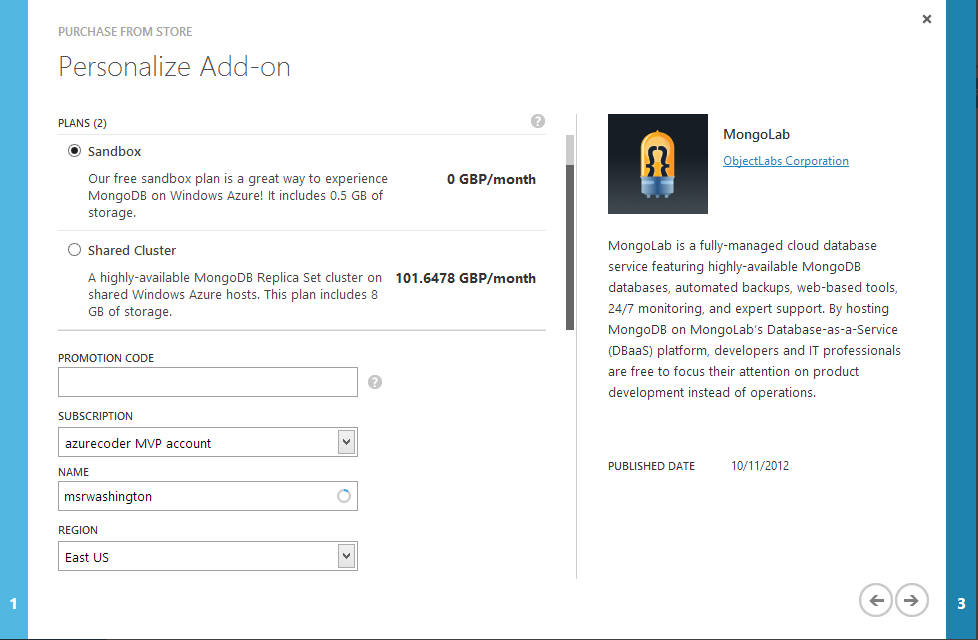
Click on purchase add-ons in the results pane.



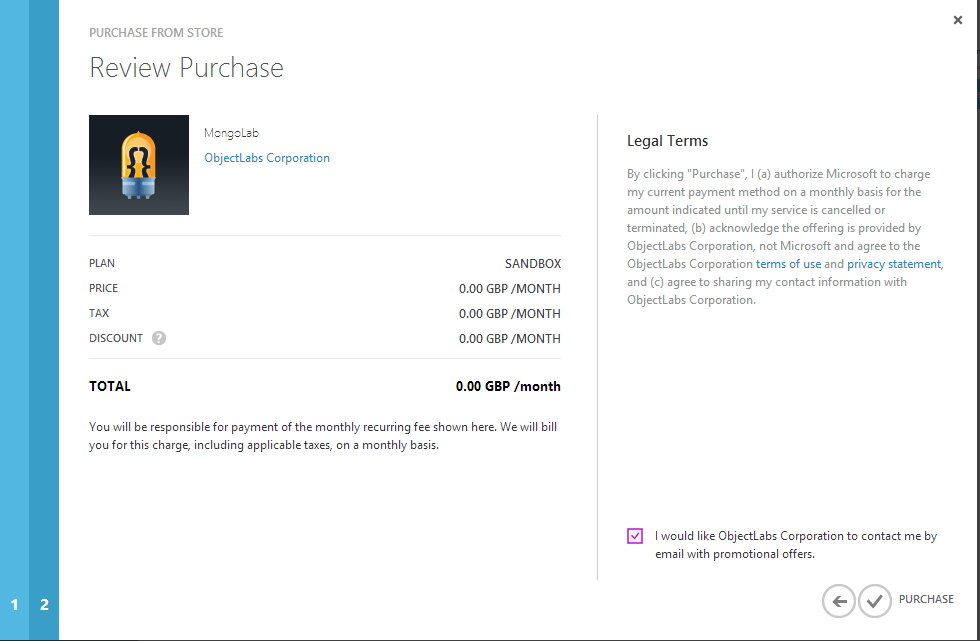
Follow the first step of the wizard:



Enter the information in the second step.



Opting for the sandbox. This is not available for EA subscriptions and in all regions.



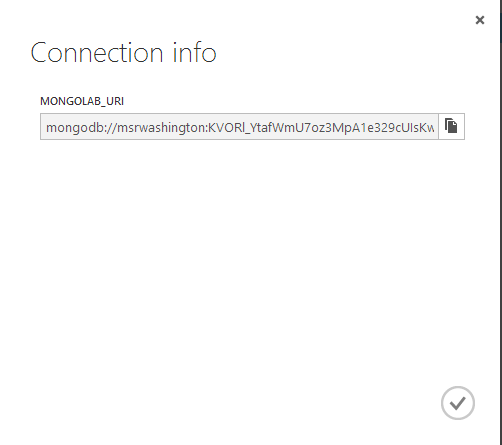
Finish as above by agreeing to the terms.

In the results pane you should see MongoDb creating.



When this is complete click on the **connection info** at the bottom of the screen.

Copy the Mongolabs Uri to use later.



## Get the python code

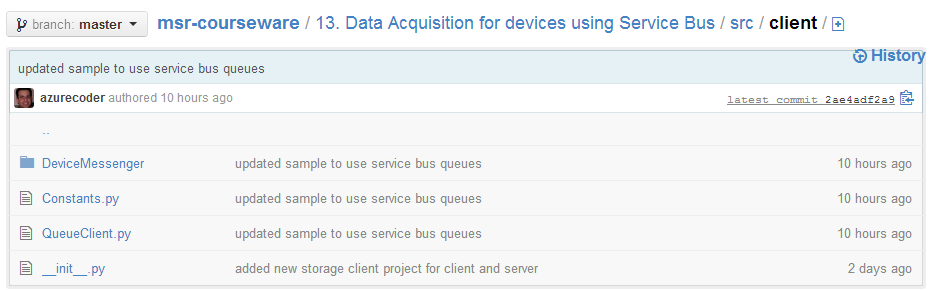
If you haven’t done so already install Python and the Azure SDK per the instructions below (covered for Windows, Linux and MacOSX):

<https://www.windowsazure.com/en-us/develop/python/common-tasks/install-python/>

Clone the git repo at:

<https://github.com/elastacloud/msr-courseware.git>

Update the constants.py file with the new information you have written down. It should be in module 13 directory/ src/client.



The file contains the following:

account\_name **=** ''

account\_key **=** ''

servicebus\_namespace **=** ""

servicebus\_issuer **=** "owner"

servicebus\_key **=** ""

Update the **account\_name** with the name of the storage account you created earlier and copy the primary key value you noted earlier to the **account\_key** constant. Update the **servicebus\_namespace** with the name of the service bus you created earlier. The **servicebus\_issuer** remains the same as **owner** and the key can be copied from the key value that was copied earlier.

## Update the server

Navigate to the src/server directory in the same root directory as above. Open the file **MSR.MessagingServer.sln** which should open up in Visual Studio.NET (2012 preferred).

Edit the web.config file after navigating to the <appSettings /> section. You should see the following values.

<add key="MsrStorageConnectionString" value="DefaultEndpointsProtocol=https;AccountName=[account name];AccountKey=[account key]” />

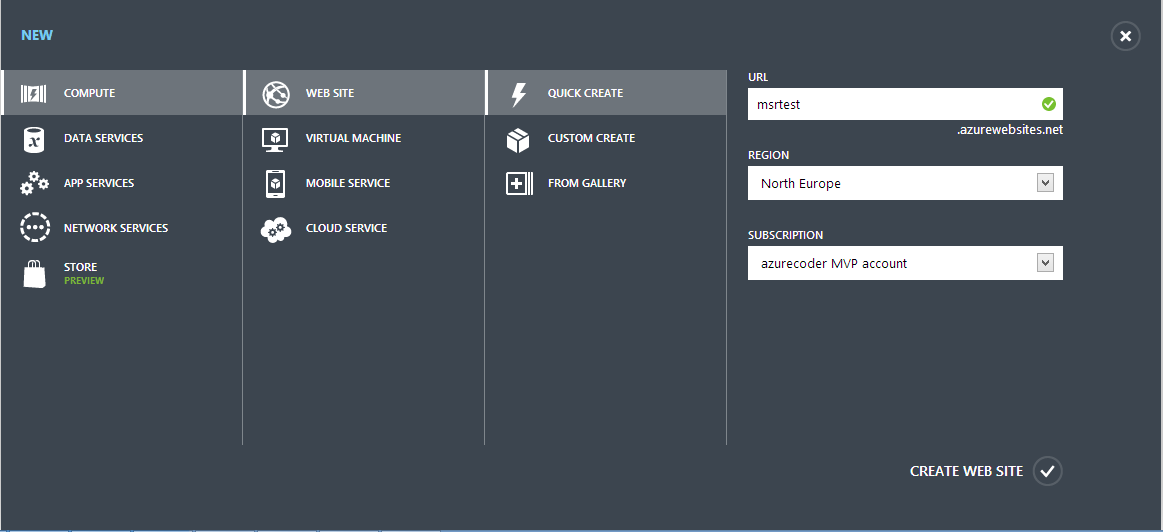
<add key="Microsoft.ServiceBus.ConnectionString" value="" />

<add key="MongoDbConnectionString" value="" />

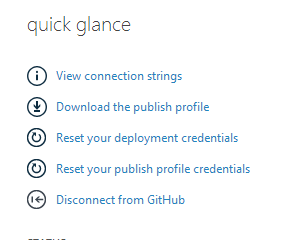
Copy the storage account name in [account name] and the [account key] that was noted earlier when the Azure Storage account was created in the key **MsrStorageConnectionString**. Add the Service Bu Connection String which was noted earlier in the **Microsoft.ServiceBus.ConnectionString** key. Add the newly created MongoDB connection string in the **MongoDbConnectionStrinng** key.

## Create a Windows Azure Website

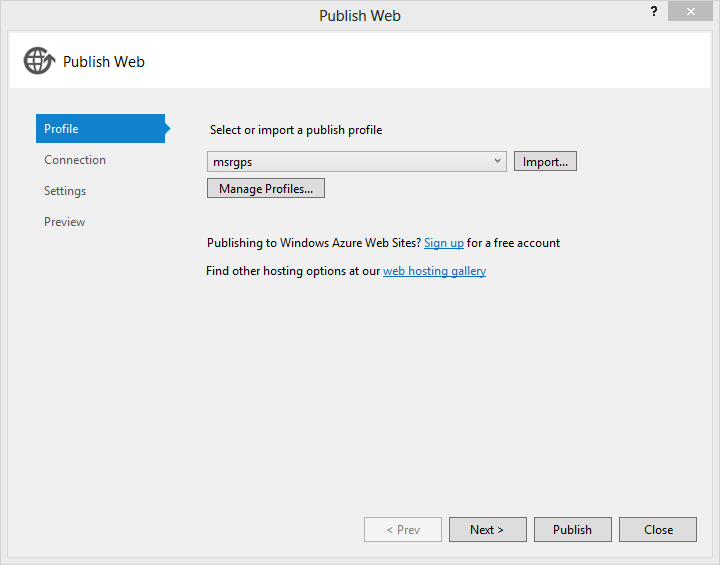
Create a Windows Azure Website in the management portal like so:



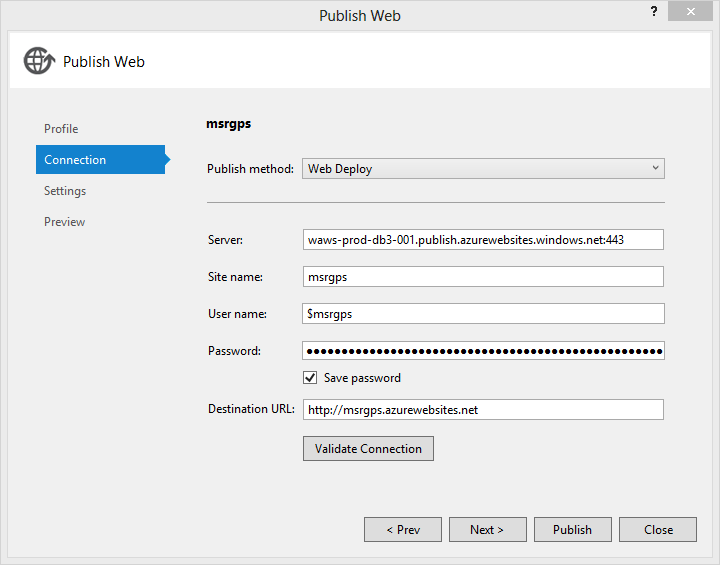
On the dashboard of the newly created website we select **Download the publish profile** from the quick glance menu. Save this on the desktop or an accessible well-known location.



In visual studio right-click on the project file **MSR.MessagingServer** and select **Publish ..**

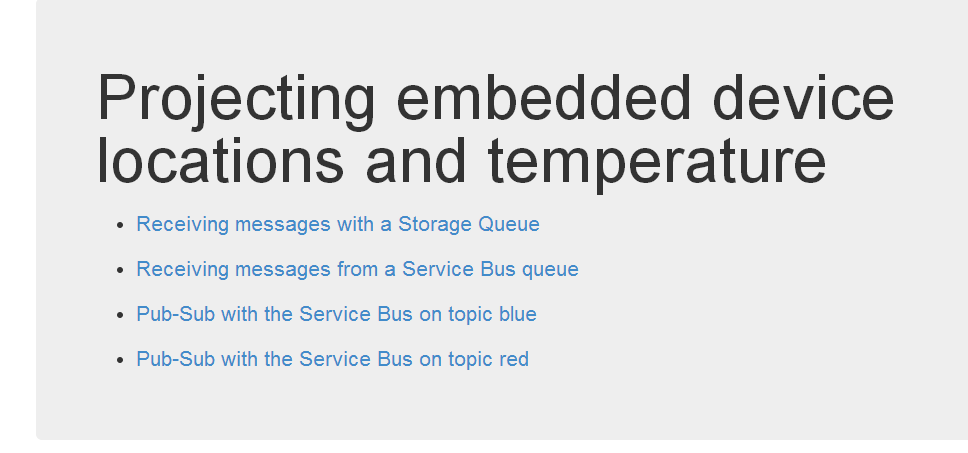


Click import and navigate to the newly downloaded Publish Profile.



Follow the wizard steps as above to publish the server project to Windows Azure. It should now be deployed and ready to use. Navigate to <http://[website> name].azurewebsites.net where [website name] is the name of the site you’ve chosen previously.

You should see an index page with the following menu.



## Running the client

The client will most likely be a Linux based machine or preferably a Raspberry Pi. Ensure Python is installed.

To install the SDK:

* curl https://raw.github.com/pypa/pip/master/contrib/get-pip.py | sudo python

Followed by:

* sudo /usr/local/bin/pip-2.7 install azure

Copy the files from the “Data Acquisition for Devices with the Service Bus” and navigate to the src/client directory.

Enter the following at the command prompt:

* QueueClient.py --type Simple --iterations 5

You should see an output similar to this 5 times:

**bcm002.queue.core.windows.net**

**/msrdevices/messages**

**[('Content-Length', '152'), ('x-ms-version', '2011-08-18'), ('x-ms-date', 'Sat, 07 Sep 2013 12:44:40 GMT'), ('Authorization', 'SharedKey bcm002:edacB4kR1Qye2KxJim+zrXBFDvF9ulUtole2fgQQNUY=')]**

**<QueueMessage> <MessageText>eyJsYXQiOiA1Ny4wNzc4ODk5Nzc1MDc3OTUsICJsbmciOiAtNy4wODYzMzA5NzAyNTkwNzQsICJ0**

**ZW1wIjogMTl9**

**</MessageText> </QueueMessage>**

**Sending message payload ZXlKc1lYUWlPaUExTnk0d056YzRPRGs1TnpjMU1EYzNPVFVzSUNKc2JtY2lPaUF0Tnk0d09EWXpN**

**ekE1TnpBeU5Ua3dOelFzSUNKMApaVzF3SWpvZ01UbDkK**

## Checking the server

From the above web menu navigate to the first item in the list.

You should see 3 points on the map. Roll over the points and see the temperature. You can relate this back to the JSON message sent by the client. Also point out above that the payload has to be Base64 encoded.

You should see 2 more points appear in front of your eyes. Run the simulation again from the client and watch another 5 points appear on the map totalling ten points.



Now we can run another command on the client to ensure that the Service Bus messages get sent to the application.

* QueueClient.py --type Messaging --iterations 5

Navigate to the Service Bus queue page and illustrate. Show the output trace and the JSON that is produced.

* QueueClient.py –colour blue --type topic --iterations 5

Issue the above command on the client and check the web page to see whether the blue topic messages have been received by checking for 5 points on the map. Check the red topics webpage to ensure that they have only been delivered to blue.

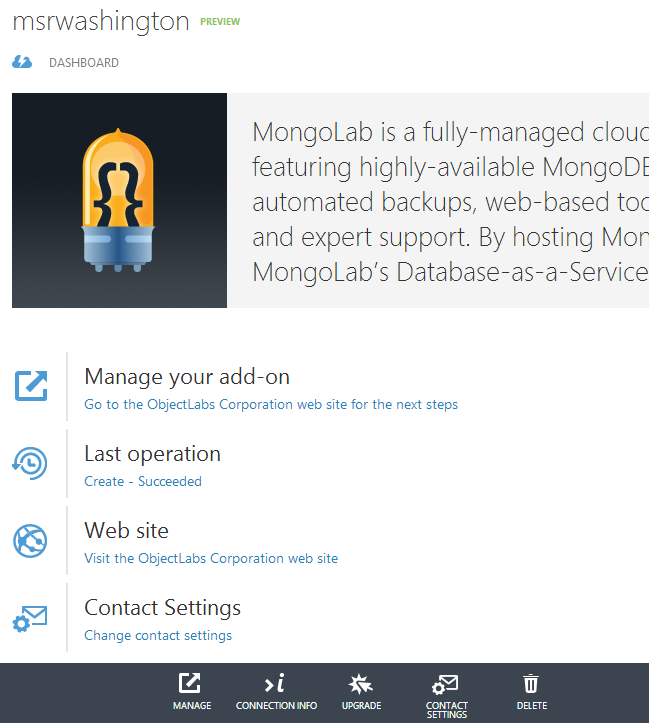
Run the same command and repeat for Red.

* QueueClient.py –colour red --type topic --iterations 5

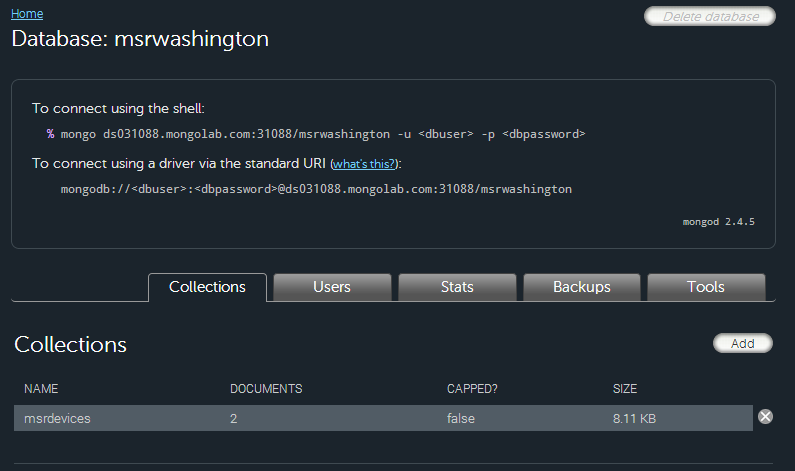
## Checking MongoDB

Navigate to the Windows Azure portal and click on the Add ons tab.

Find the Mongolabs instance we created earlier and navigate to the welcome page.



At the bottom click on manage. You should screen like this.



Double-click on the msrdevices collection. You should several records similar to this showing the JSON data and the special unique MongoDb ID field. Explain how Mongo has just taken and persisted the JSON and can scale and provide a petabyte storage mechanism for device messages for audit.

**{**

**"\_id": {**

**"$oid": "522a60c6564b39b5f794cc95"**

**},**

**"Lat": 55.9228613186053,**

**"Long": -6.83752333883609,**

**"Temp": 17**

**}**