# **Microsoft Innovation Centre**

# **(Jordan)**

### Clinic System

Date: 31-12-2013

Developed at: Microsoft innovation Centre (Jordan).

By: Zainab Al-Haidary

### Technical Guide

### Introduction:

This document represent a technical guide that helps the clinic IT team to set up an Azure account for the application to save the clinics data.

### Steps:

1. If you do not have a Windows Azure subscription already, then sign up for a [free trial](http://go.microsoft.com/fwlink/p/?linkid=286807) .
2. Install the [command line tools](http://go.microsoft.com/fwlink/p/?linkid=275464) for Windows Azure.
3. Download the credentials needed to talk to Windows Azure. You need to do this once to manage all subsequent commands to Mobile Services.

To do this:

* + Download the Windows Azure management credentials by entering the command **azure account download**. This will open up a web page to login to the [Management Portal](http://go.microsoft.com/fwlink/p/?LinkID=306562) for Windows Azure. Once you log in, it will generate and prompt you to download a publishsettings file for your Windows Azure subscription. Save this file to a location on your machine.
  + Import the publishsettings file from the saved location. This will configure your command line client to manage all your Windows Azure services from the command line. To do this, enter the command:

**azure account import [SaveLocation]**

1. Create a Windows Azure Mobile Service by entering the command:

**azure mobile create [AzureMobileServiceName] [sqlAdminUsername] [sqlAdminPassword]**

1. Setup the tables and set the permissions for the table operations for your Windows Azure Mobile Service by entering the following commands:

**azure mobile table create [AzureMobileServiceName] Appointment**

**azure mobile table create [AzureMobileServiceName] Device**

**azure mobile table create [AzureMobileServiceName] Drug**

**azure mobile table create [AzureMobileServiceName] Invitation**

**azure mobile table create [AzureMobileServiceName] patient**

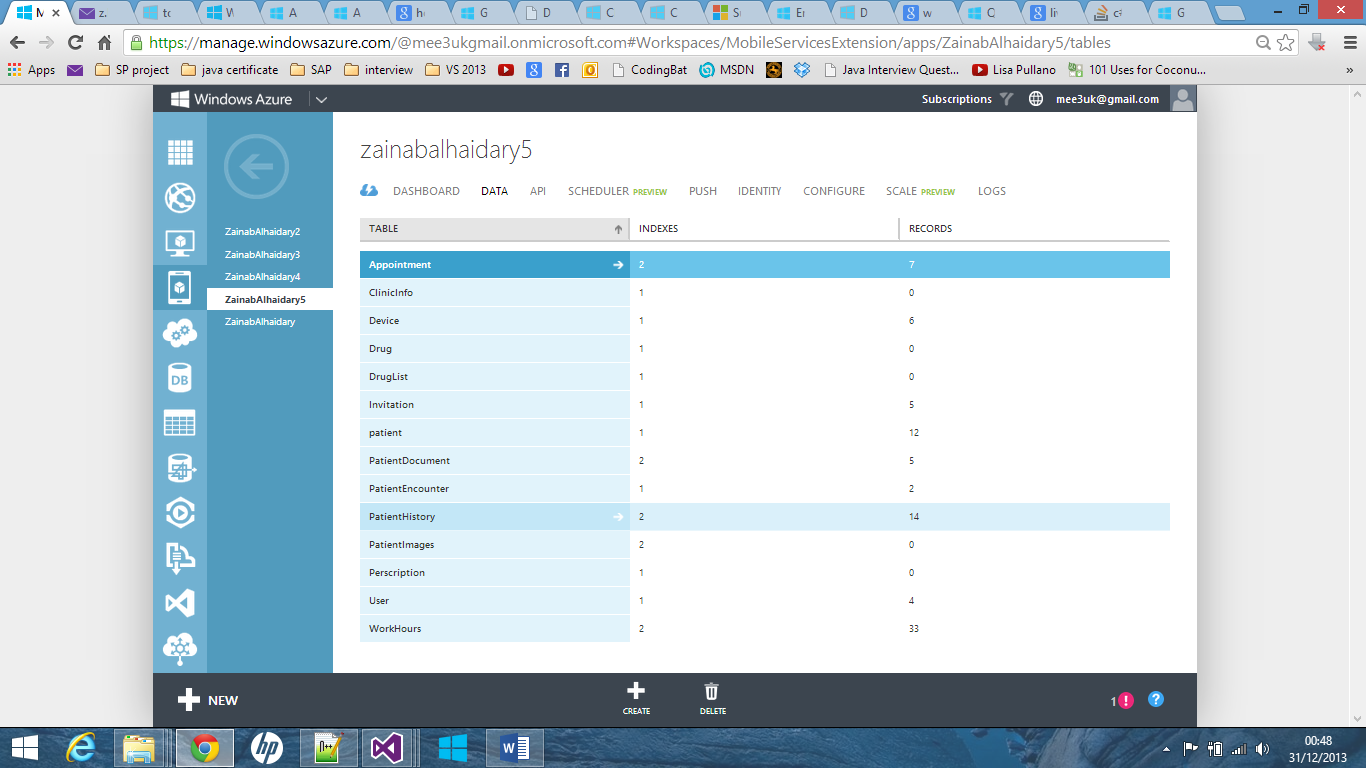
**azure mobile table create [AzureMobileServiceName] PatientDocument**

**azure mobile table create [AzureMobileServiceName] PatientEncounter**

**azure mobile table create [AzureMobileServiceName] PatientHistory**

**azure mobile table create [AzureMobileServiceName] User**

**azure mobile table create [AzureMobileServiceName] WorkHours**



1. Upload the scripts to your Windows Azure Mobile Service which will set up the database. In your browser open [www.windowsazure.com](http://www.windowsazure.com) and log into your azure account. Select your **[AzureMobileServiceName]** then click on **Data** tab to see the tables. Select **Device** table, remove the old **Insert** script and paste the following:

var HttpStatusOk = 200;

var HttpStatusBadRequest = 400;

function insert(item, user, request) {

// We don't trust the client, we always set the user on the server

//item.userId = user.userId;

// Require an installationId

if (!item.installationId || item.installationId.length === 0) {

request.respond(HttpStatusBadRequest, "InstallationId is required");

return;

}

// Find any records that match this device already (user and installationId combo)

var device = tables.getTable('device');

device.where({

userId: item.userId,

installationId: item.installationId

}).read({

success: function (results) {

if (results.length == 1) {

// This device already exists, so don't insert the new entry,

// update the channelUri (if it's different)

if (item.channelUri === results[0].channelUri) {

request.respond(HttpStatusOk, results[0]);

return;

}

// Otherwise, update the notification id

results[0].channelUri = item.channelUri;

device.update(results[0], {

success: function () {

request.respond(HttpStatusOk, results[0]);

return;

}

});

} else {

request.execute();

}

}

});

}

Then go to the **patient** table and replace the **Insert** script with the following:

var azure = require('azure');

var qs = require('querystring');

var appSettings = require('mobileservice-config').appSettings;

var HttpStatusOk = 200;

var HttpStatusBadRequest = 400;

var HttpStatusForbidden = 403;

var HttpStatusNotFound = 404;

//var devices = tables.getTable('device');

function insert(item, user, request) {

console.log("user is ", user);

// Get storage account settings from app settings.

var accountName = appSettings.STORAGE\_ACCOUNT\_NAME;

var accountKey = appSettings.STORAGE\_ACCOUNT\_ACCESS\_KEY;

var host = accountName + '.blob.core.windows.net';

if ((typeof item.containerName !== "undefined") && (

item.containerName !== null)) {

// Set the BLOB store container name on the item, which must be lowercase.

item.containerName = item.containerName.toLowerCase();

// If it does not already exist, create the container

// with public read access for blobs.

var blobService = azure.createBlobService(accountName, accountKey, host);

blobService.createContainerIfNotExists(item.containerName, {

publicAccessLevel: 'blob'

}, function(error) {

if (!error) {

// Provide write access to the container for the next 5 mins.

var sharedAccessPolicy = {

AccessPolicy: {

Permissions: azure.Constants.BlobConstants.SharedAccessPermissions.WRITE,

Expiry: new Date(new Date().getTime() + 5 \* 60 \* 1000)

}

};

// Generate the upload URL with SAS for the new image.

var sasQueryUrl =

blobService.generateSharedAccessSignature(item.containerName,

item.resourceName, sharedAccessPolicy);

// Set the query string.

item.sasQueryString = qs.stringify(sasQueryUrl.queryString);

// Set the full path on the new new item,

// which is used for data binding on the client.

item.imageUri = sasQueryUrl.baseUrl + sasQueryUrl.path;

} else {

console.error(error);

}

request.execute();

sendNotifications();

});

} else {

request.execute();

sendNotifications();

}

function sendNotifications() {

var channelsTable = tables.getTable('Device');

channelsTable.read({

success: function (devices) {

devices.forEach(function (device) {

push.wns.sendToastText04(device.channelUri, {

text1: 'A new patient has been added',

text2: item.fname

}, {

success: function (pushResponse) {

console.log("Sent push:", pushResponse);

}

});

});

}

});

}

}

Go to the **PatientDocument** table and replace the **Insert** script with the following:

var azure = require('azure');

var qs = require('querystring');

var appSettings = require('mobileservice-config').appSettings;

var HttpStatusOk = 200;

var HttpStatusBadRequest = 400;

var HttpStatusForbidden = 403;

var HttpStatusNotFound = 404;

//var devices = tables.getTable('device');

function insert(item, user, request) {

console.log("user is ", user);

// Get storage account settings from app settings.

var accountName = appSettings.STORAGE\_ACCOUNT\_NAME;

var accountKey = appSettings.STORAGE\_ACCOUNT\_ACCESS\_KEY;

var host = accountName + '.blob.core.windows.net';

if ((typeof item.containerName !== "undefined") && (

item.containerName !== null)) {

// Set the BLOB store container name on the item, which must be lowercase.

item.containerName = item.containerName.toLowerCase();

// If it does not already exist, create the container

// with public read access for blobs.

var blobService = azure.createBlobService(accountName, accountKey, host);

blobService.createContainerIfNotExists(item.containerName, {

publicAccessLevel: 'blob'

}, function (error) {

if (!error) {

// Provide write access to the container for the next 5 mins.

var sharedAccessPolicy = {

AccessPolicy: {

Permissions: azure.Constants.BlobConstants.SharedAccessPermissions.WRITE,

Expiry: new Date(new Date().getTime() + 5 \* 60 \* 1000)

}

};

// Generate the upload URL with SAS for the new image.

var sasQueryUrl =

blobService.generateSharedAccessSignature(item.containerName,

item.resourceName, sharedAccessPolicy);

// Set the query string.

item.sasQueryString = qs.stringify(sasQueryUrl.queryString);

// Set the full path on the new new item,

// which is used for data binding on the client.

item.imageUri = sasQueryUrl.baseUrl + sasQueryUrl.path;

} else {

console.error(error);

}

request.execute();

});

} else {

request.execute();

}

}

Go to the **User** table and eplace the **Insert** script with the following:

function insert(item, user, request) {

request.execute();

sendNotifications();

function sendNotifications() {

var channelsTable = tables.getTable('Device');

channelsTable.read({

success: function (devices) {

devices.forEach(function (device) {

push.wns.sendToastText04(device.channelUri, {

text1: 'A new user has been added',

text2: item.fname

}, {

success: function (pushResponse) {

console.log("Sent push on user creation:", pushResponse);

}

});

});

}

});

}

}

Go to the **WorkHours** table and replace the **Insert** script with the following:

function insert(item, user, request) {

var User = tables.getTable('User');

User.where({

UserId: user.userId

}).read({

success: function (results) {

if (results.length == 1) {

console.log("user admin value is ",results[0].isadmin)

if(results[0].isadmin){

request.execute();

}

else{

request.respond(statusCodes.FORBIDDEN, 'You are not authorized');

}

}

}

});

//console.log("user is ", User);

//console.log( "original ",user)

}

### Configuring the Client Secret and Package Security Indentifier

1. Configure your Windows Azure Mobile Service with the **Client secret** and **Package security identifier**

**azure mobile config set [AzureMobileServiceName] microsoftAccountClientId xxxxxxxxxxxxxxxxxxxxx**

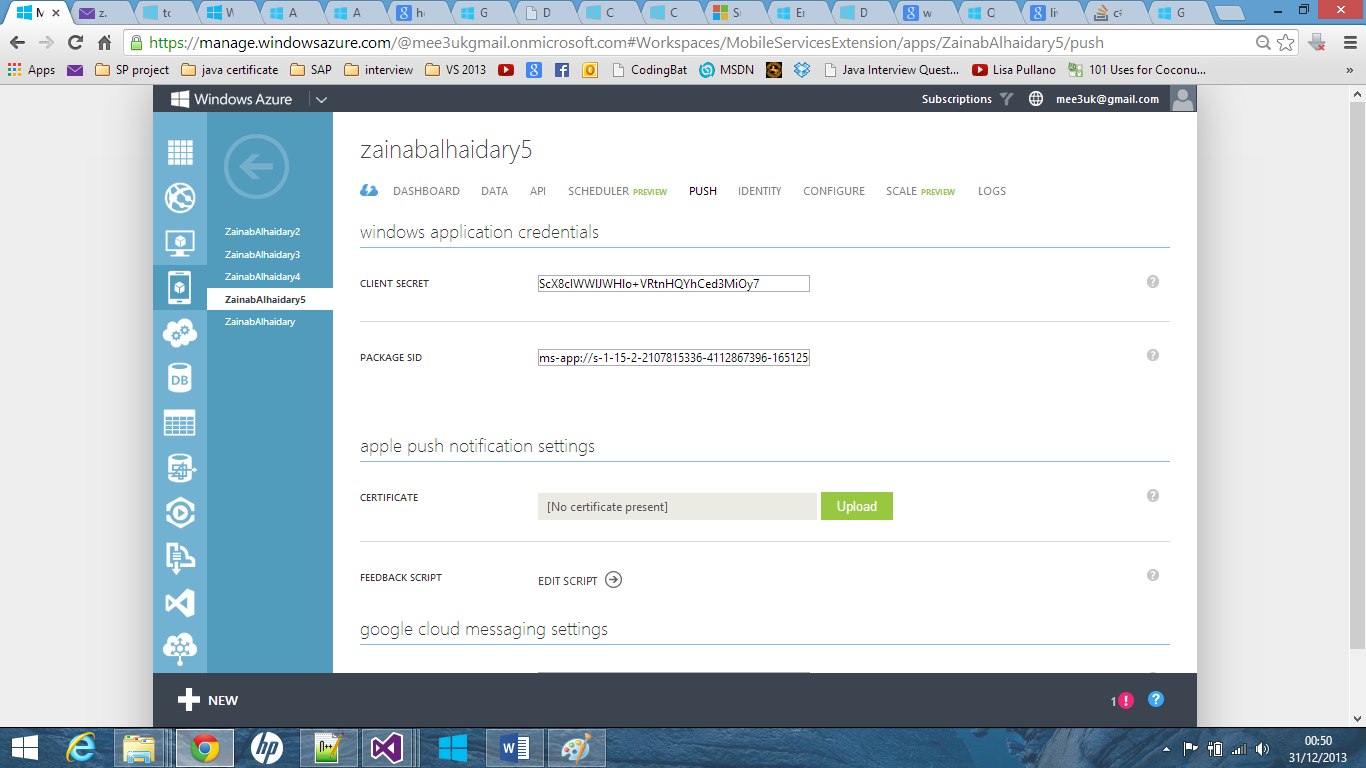
**azure mobile config set [AzureMobileServiceName] microsoftAccountClientSecret xxxxxxxxxxxxxxxxxxxxxxx**

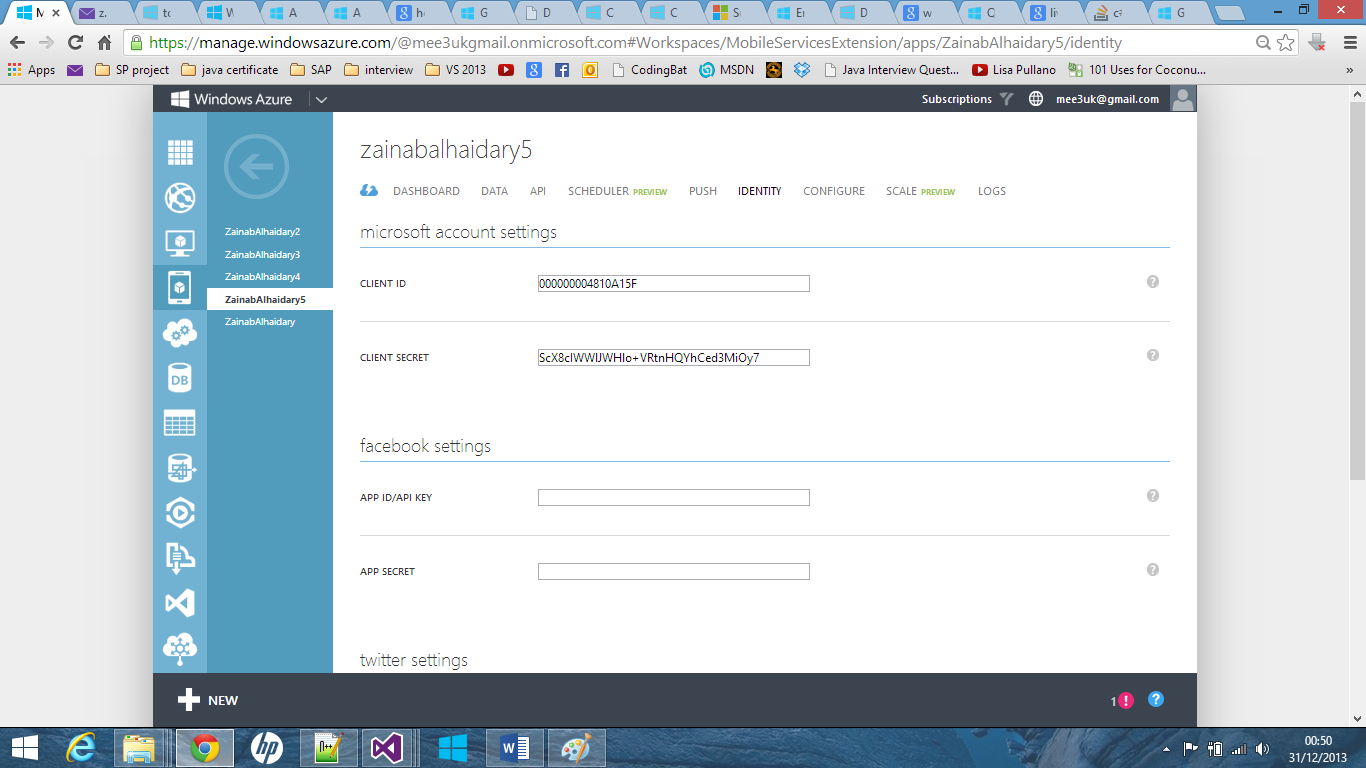
**azure mobile config set [AzureMobileServiceName] microsoftAccountPackageSID ms-app://s-1-15-2-2107815336-4112867396-1651256795-2060734397-2986304955-491198237-3132839075**

1. Get the ApplicationUrl and ApplicationKey for your Windows Azure Mobile Service:

**azure mobile show [AzureMobileServiceName]**

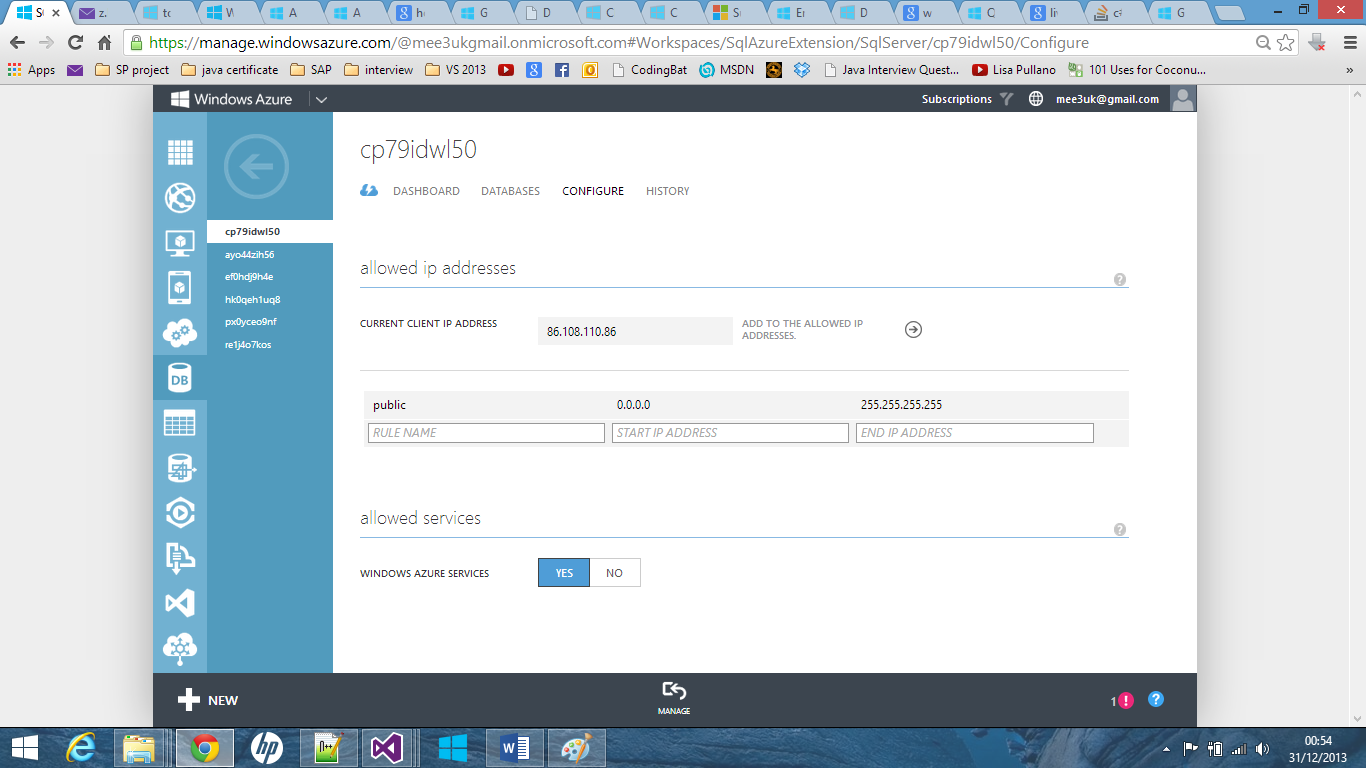
These are the value that you should insert in the registration page which is the first page of the program.





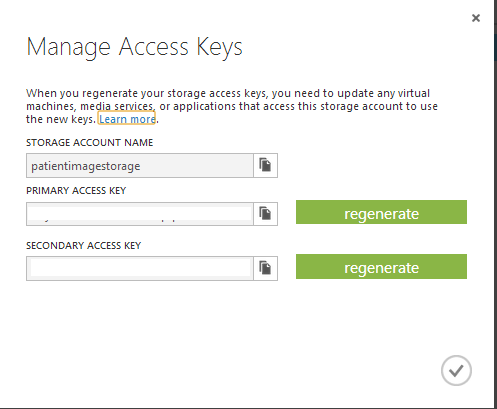
### Server Allowed IP addresses:

Modify the server allowed ip address to include all the machines in your clinic.



### Create Blob storage account:

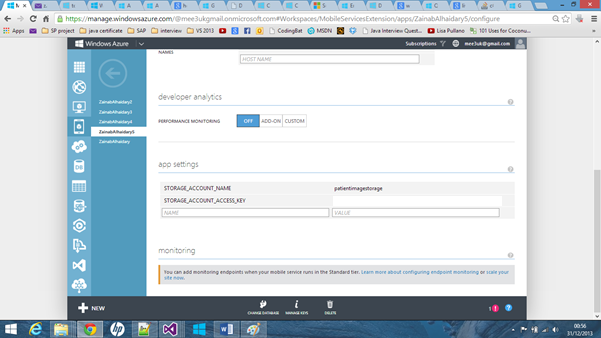
1. Follow the following tutorial to create the [storage account](http://www.windowsazure.com/en-us/manage/services/storage/how-to-create-a-storage-account/) and name it **patientimagestorage**
2. In the Management Portal, click **Storage**, click the storage account, then click **Manage Keys**.
3. Make a note of the **Storage Account Name** and **Access Key**.

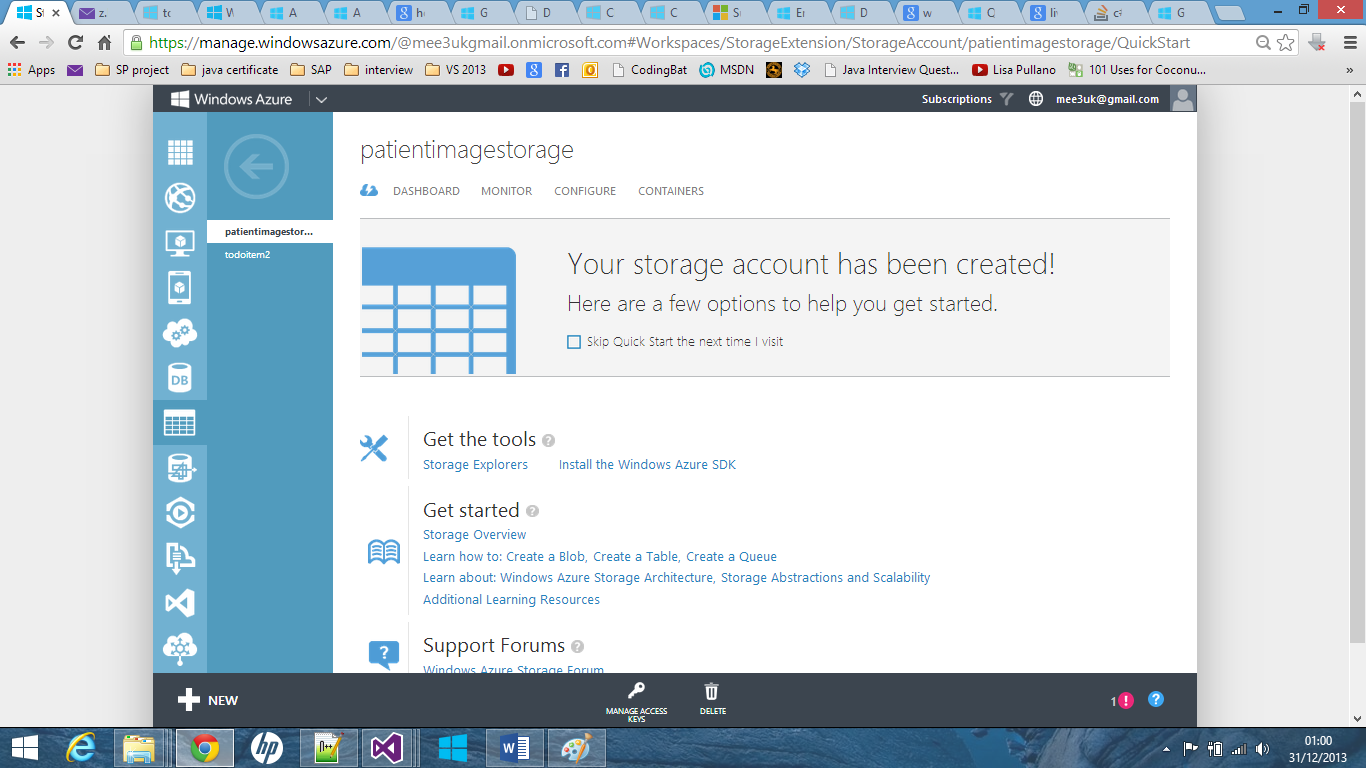


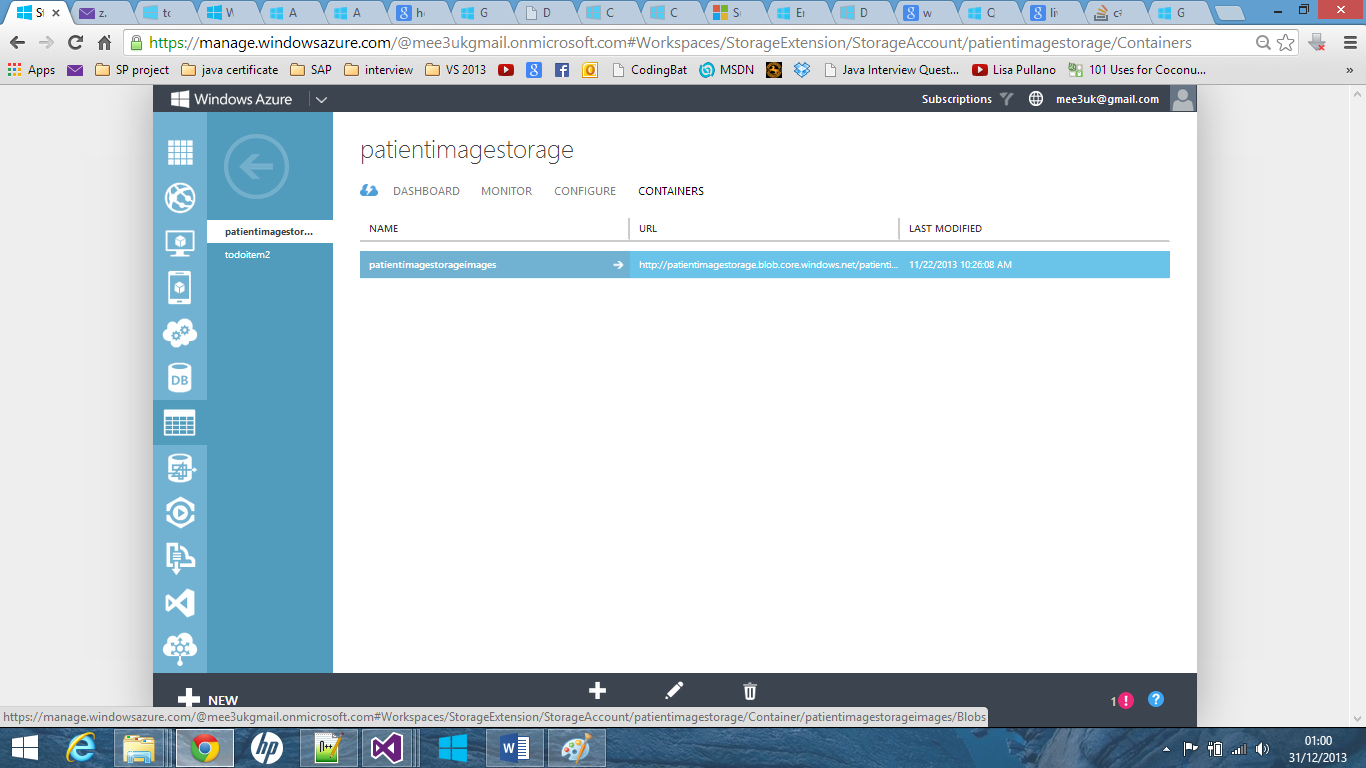
1. In your mobile service, click the Configure tab, scroll down to App settings and enter a Name and Value pair for each of the following that you obtained from the storage account, then click Save.

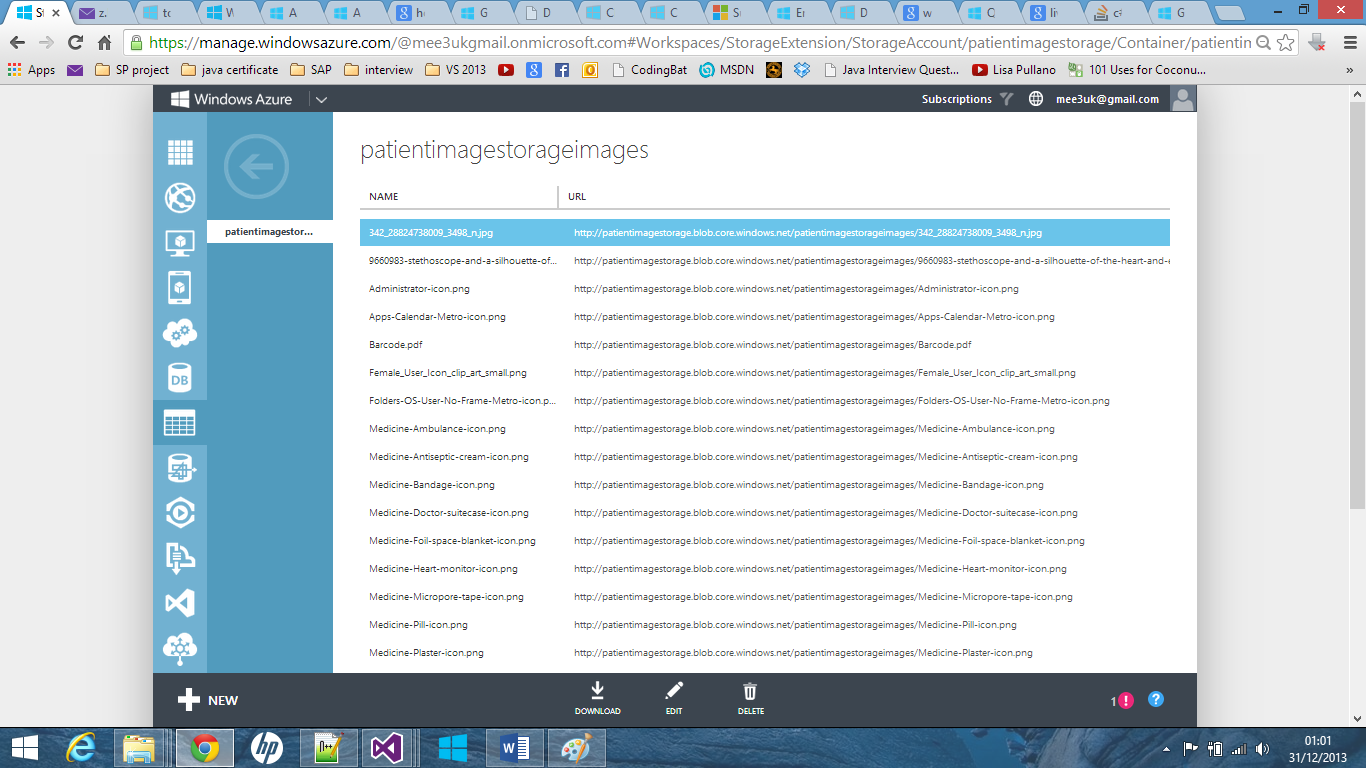
STORAGE\_ACCOUNT\_NAME

STORAGE\_ACCOUNT\_ACCESS\_KEY



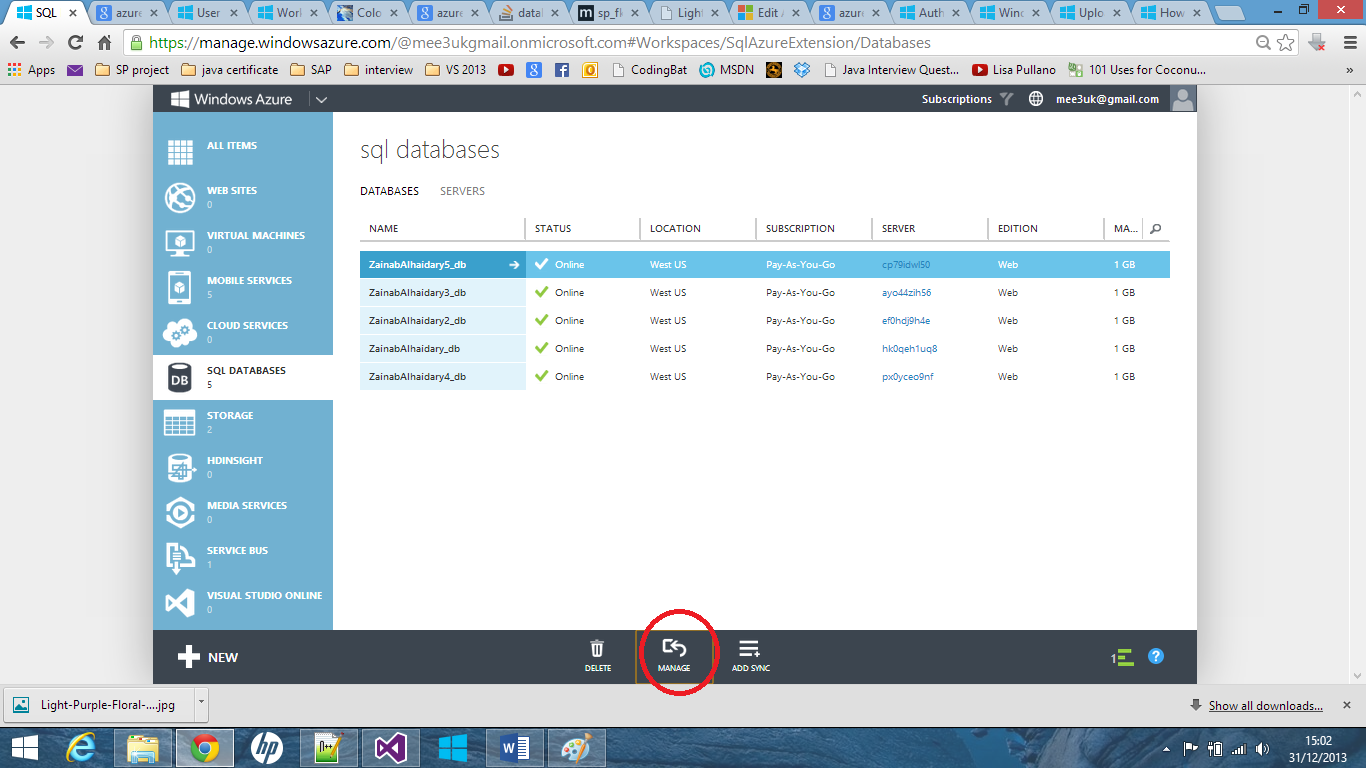




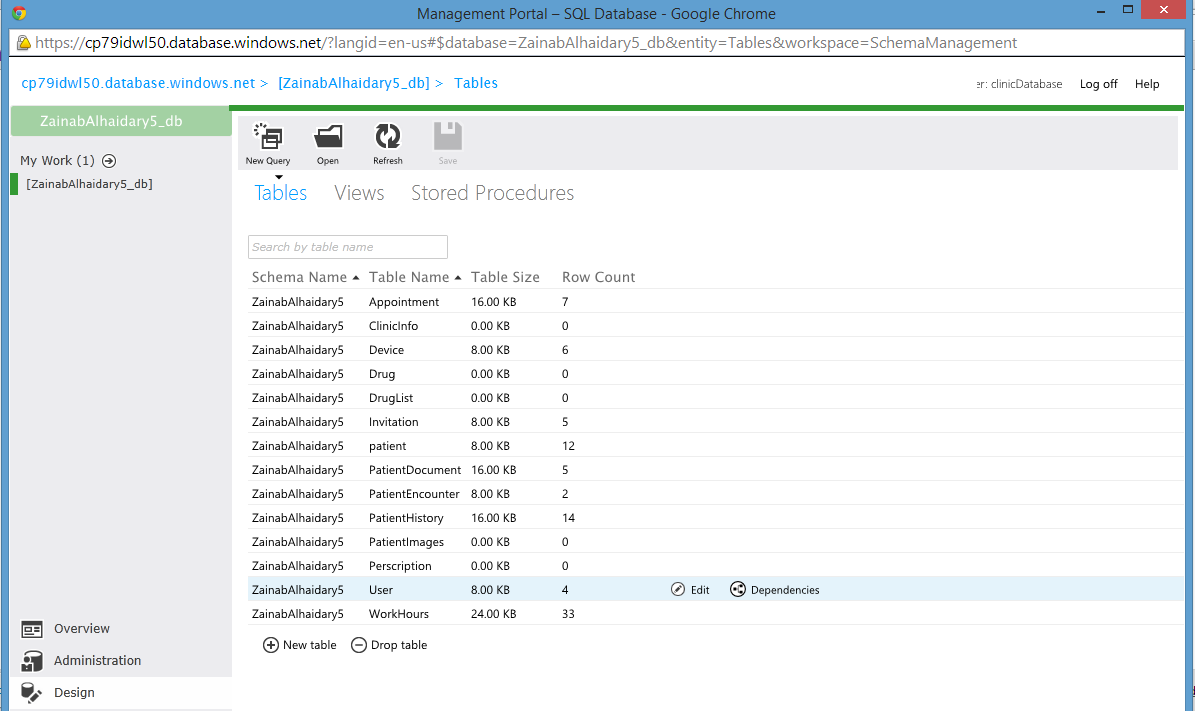


### Make a certain user an Admin:

1. After running the application at least one time. Sign-in to your azure account.
2. Go to the SQL Database section and click on **Manage.**



1. Insert your credentials.
2. Click on **Design** on the left side.
3. Click on the **user** table and click edit:



1. Click on **Data** and change the **isadmin** field of the desired user to **true.** Click save and exit.