Optical Character Recognition using Microsoft Azure Cognitive Services

Step-By-Step Instructions

IoT Cart

Contents

[Change Log 2](#_Toc38013768)

[Purpose 3](#_Toc38013769)

[Azure Services Used 3](#_Toc38013770)

[Required Hardware 3](#_Toc38013771)

[Step-by-Step Instructions 3](#_Toc38013772)

[Choose a device 3](#_Toc38013773)

[Anaconda Configuration 3](#_Toc38013774)

[Create Resource Group (optional) 4](#_Toc38013775)

[Create “Azure SQL” Resource 5](#_Toc38013776)

[SQL Connection String and Successfully Connecting to the DB 10](#_Toc38013777)

[Configure Database Compute and Storage 11](#_Toc38013778)

[Create Server (optional) 12](#_Toc38013779)

[Create Compute Vision Resource 13](#_Toc38013780)

[Computer Vision Key and Endpoint 15](#_Toc38013781)

[Ensure SQL DB Tables Exist 15](#_Toc38013782)

[Setup the Camera Scene 15](#_Toc38013783)

[Execute the Scenario 16](#_Toc38013784)

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

© 2020 Microsoft. All rights reserved. This document is for informational purposes only. Microsoft makes no warranties, express or implied, with respect to the information presented here.

# Change Log

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Update** |
| 4/17/2020 | Josh Heitzman | First Draft |

# Purpose

This document will show a basic demonstration of:

1. the text recognition capabilities of the Computer Vision resource included in the Azure Cognitive Services suite,
2. and storing data in Azure SQL.

# Azure Services Used

1. Azure Cognitive Services: Computer Vision
2. Azure SQL DB

# Required Hardware

This scenario requires the use of an OpenCV compatible camera (typically most USB cameras) attached to a device running Windows or Linux that has a local copy of the code for this scenario.

# Step-by-Step Instructions

## Choose a device

The Windows PC can be utilized to run this scenario, but it will also work on any device with an OS that support a USB camera and the Anaconda distribution of Python.

## Anaconda Configuration

Install Anaconda 2020.02 64-bit from <https://repo.anaconda.com/archive/>. Feel free to try a newer version, but its possible it will not work. This installation of Anaconda will be referred to as ‘your Anaconda environment’ in the following instructions.

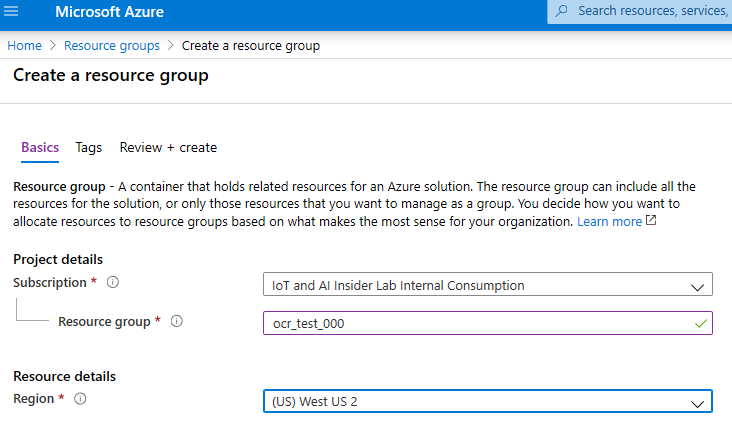
In your Anaconda environment run the following commands:

pip install azure-cognitiveservices-vision-computervision==0.5.0 msrest==0.6.13

## Create Resource Group (optional)

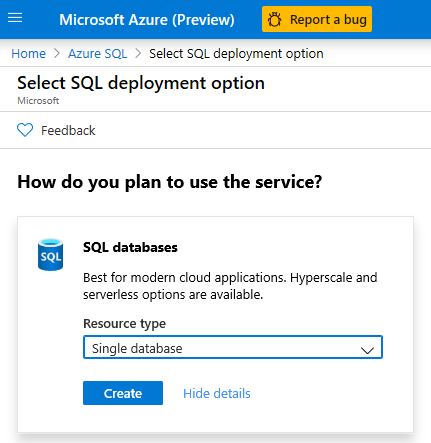
Feel free to use an existing resource group rather than creating a new one.

* Navigate to “Resource groups” and press the “Add” button.
* On the “Create a resource group” page, ensure you have selected the correct subscription and region to create your resource group under.
* Enter a name in text box for “Resource group” such as ‘ocr\_test\_000’.
* Press the “Review + create” button.
* Press the “Create” button on the next page.

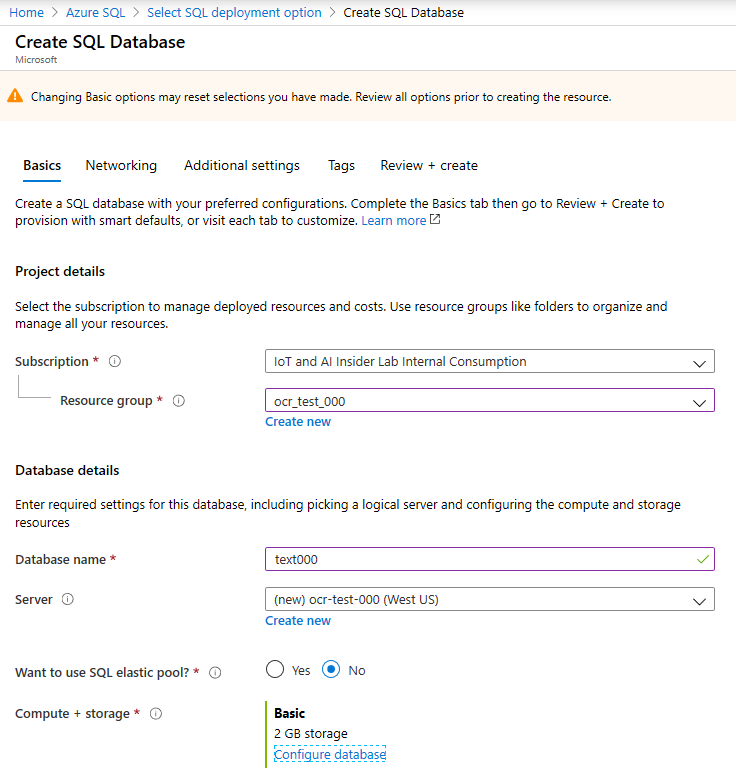


## Create “Azure SQL” Resource

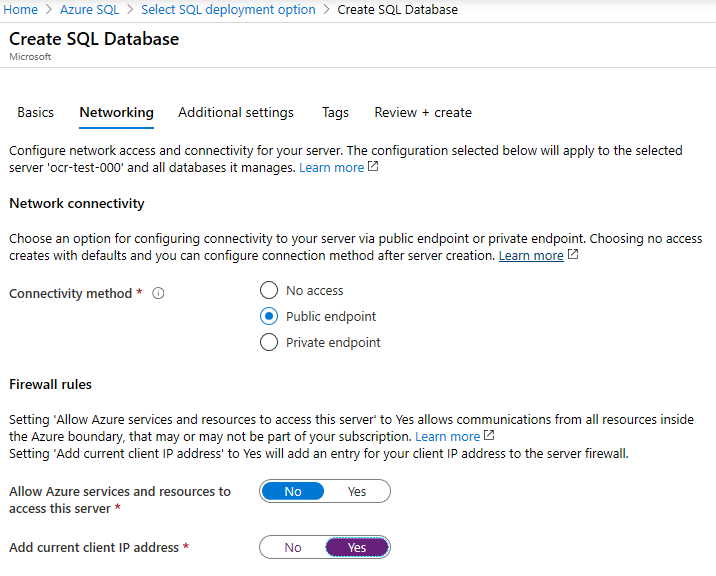
* Navigate to “Azure SQL” and press the “Add” button.
* On the “SQL databases” tile select “Single database”. Feel free to use a different option, but this should be the cheapest option.
* Press the “Create” button on the “SQL databases” tile.



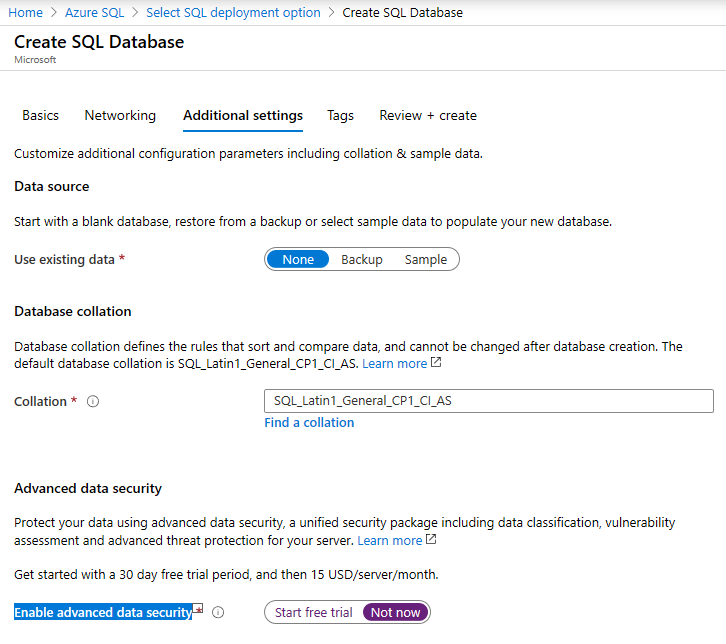
* Ensure the correct resource group is selected.
* Enter a database name.
* Feel free to use an existing server or see the section “Create Server” for creating a new server.
* Press the Configure database link and configure it to “Basic” with 5 DTUs as per section “Configure Database Compute and Storage”.
* Click the “Next: Networking” button rather than “Review + create”.



* Set the “Connectivity method” to “Public endpoint”.
* Under “Firewall rules” set “Add current client IP address” to “Yes”.
* Press “Next: Additional settings”



* Set “Enable advanced data security” to “Not now”.
* Press “Review + Create”



Wait for the deployment to complete and then press the “Go to resource” button.

### SQL Connection String and Successfully Connecting to the DB

* In the left-hand pane for the SQL select “Connection strings”.
* Select the “ODBC” tab.
* Copy the connection string from the overview page for the SQL DB.
* Replace the substring “{your\_password\_here}” with the actual password.

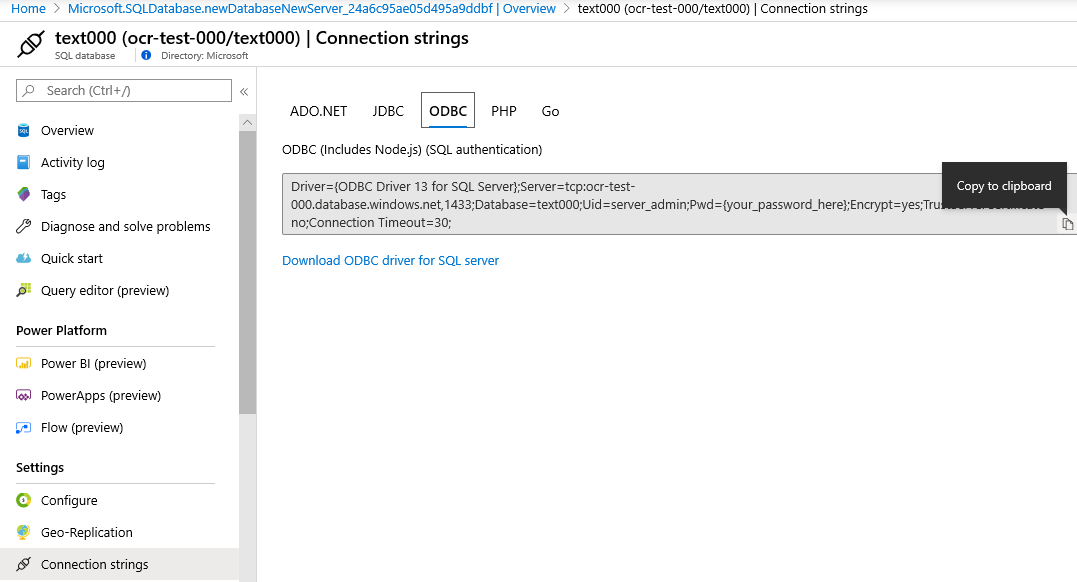
Make a note of the string as it is required to run the scripts were the command includes:

<SQL connection string>

You may also find it necessary to change the string for the driver parameter to ‘{ODBC Driver 17 for SQL Server}’ or something else as it is dependent on what ODBC drivers you have installed locally. See the following for more info:

* <https://stackoverflow.com/questions/46045834/pyodbc-data-source-name-not-found-and-no-default-driver-specified>
* <https://docs.microsoft.com/en-us/sql/connect/odbc/windows/system-requirements-installation-and-driver-files?view=sql-server-ver15#installing-microsoft-odbc-driver-for-sql-server>

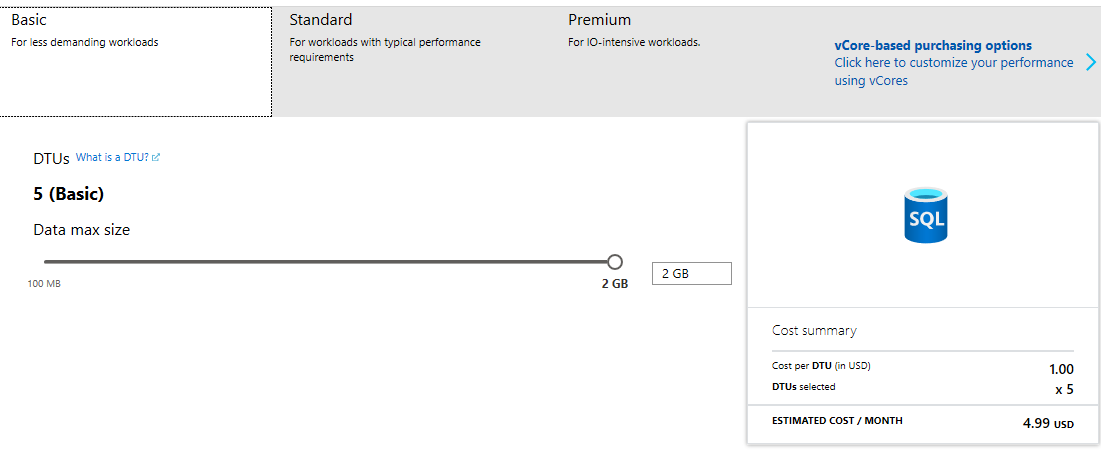
You may also find that connecting fails and this may be due too you’re IP address not being whitelisted in the firewall. The most straightforward resolution to this is to utilize Microsoft SQL Server Management Studio to connect to the SQL server, as it can modify the firewall. If the SQL can’t be connected to via Microsoft SQL Server Management Studio the Python code will not be able to connect to it either.



### Configure Database Compute and Storage

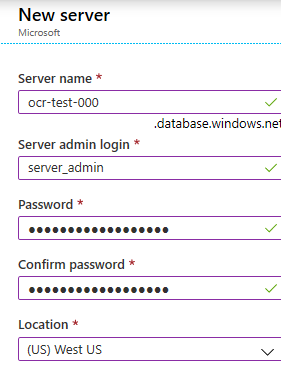
It will likely default to “General Purpose”, but you should see a link “Looking for basic, standard, premium” that will allow you to configure it to Basic with 5 DTUs.

Hit the “Apply” button once the page looks like the following.



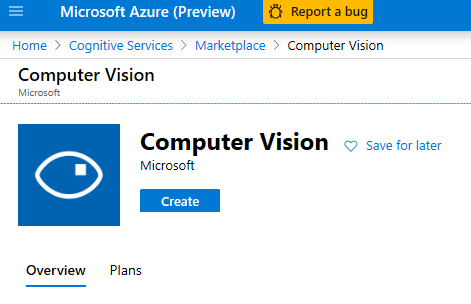
### Create Server (optional)

* Go to the “New server” bar that pops up on the right-hand side when the “Create new” link is pressed under the “Server” drop down box.
* Enter values for all fields in the bar. In the author’s case the US West 2 region didn’t allow creating a server, so the US West region was used instead.

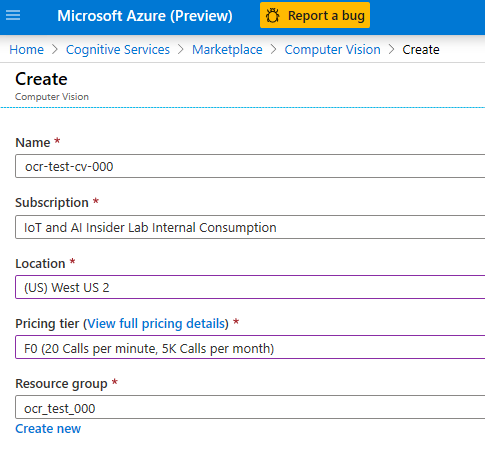


## Create Compute Vision Resource

* Navigate to “Cognitive Services” and press the “Add” button.
* On the “Marketplace” page locate the tile titled “Computer Vision” and press it.
* On the “Compute Vision” page press the “Create” button.



* On the “Create” page ensure the correct subscription, location, and resource group are set.
* You likely want to set the “Pricing tier” to the cheapest option available, which may be the “F0” tier.
* Press the “Create” button.



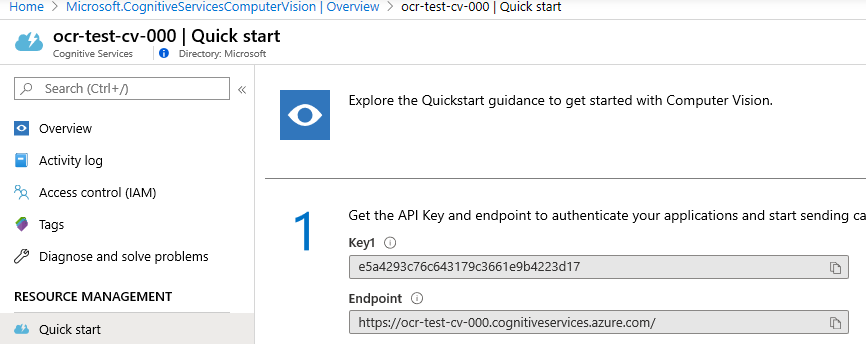
Wait for the deployment to complete and then press the “Go to resource” button.

### Computer Vision Key and Endpoint

Make a note of the values for both Key1 and Endpoint as they are required where the following tokens are present in the scenario script command:

<Key1>

<Endpoint>



## Ensure SQL DB Tables Exist

A script is provided at <repo root>/cv/ocr/sql/tools/setup.py that will create any SQL tables that are missing. The script will not fix up tables that exist and have the wrong schema (i.e. the table needs to be deleted before running the script).

In your Anaconda environment run the following commands:

cd <repo root>/cv/ocr/sql/tools

python setup.py --connection\_string "<SQL connection string>"

## Setup the Camera Scene

Ensure the USB camera is plugged into the device on which the scripts will be run.

A script is provided at <repo root>/cv/ocr/tools/camera\_viewer.py that will create any SQL tables that are missing. The script will not fix up tables that exist and have the wrong schema (i.e. the table needs to be deleted before running the script).

In your Anaconda environment run the following commands:

cd <repo root>/cv/ocr/tools

python camera\_viewer.py

You should see a window pop up that shows a video feed from the camera. Utilize this to position the object with the text to recognize, so that the text is clearly legible in the video feed.

## Execute the Scenario

After following successfully completing the previous steps the device and cloud services should now be configured to allow the scenario script to run successfully.

In your Anaconda environment run the following commands:

cd <repo root>/cv/ocr/tools

python capture\_recognize\_and\_store.py --subscription\_key "<Key1>" --endpoint "<Endpoint>" --connection\_string "<SQL connection string>"

The captured image is written to <repo root>/captured.png by default, but the script supports an additional parameter ‘--image\_path’ for the absolute path to where the captured image should be written the filesystem.

After running the script verify that:

1. The expect image is found at the image\_path location.
2. There is a new row in the ProcessedText SQL DB table.
3. The Text field of that row contains text matching the text on the object.