FY19 DTA AI Hack – Proctor Guide

AI-oriented services such as Azure Bot Services, Azure Search, and Cognitive Services

# Goals

Most challenges observed by customers in these realms are in stitching multiple services together. As such, where possible, we have tried to place key concepts in the context of a broader example.

Once all hackathon challenges are completed, you should be able to:

* Configure your apps to call Cognitive Services
* Build an application that calls various Cognitive Services APIs (specifically Computer Vision)
* Implement Azure Search features to provide a positive search experience inside applications
* Configure an Azure Search service to extend your data to enable full-text, language-aware search
* Build, train, and publish a LUIS model to help your bot communicate effectively
* Build an intelligent bot using Microsoft Bot Framework that leverages LUIS and Azure Search
* Effectively log chat conversations in your bot
* Perform rapid development/testing with Ngrok and test your bots with unit tests and direct bot communication
* Effectively leverage the custom vision service to create image classification services that can then be leveraged by an application

# Background Knowledge

This hackathon is meant for an AI Developer on Azure. Previous experience with the following may be helpful:

* + Azure Cognitive Services
  + Visual Studio
  + C#
  + Microsoft's Bot Framework

# Introduction

## Lecture

AI Overview, including introductions to Azure Bot Services, Cognitive Services, and Azure Search

# Getting Started

## Pre-requisites

This is a list of pre-requisites needed to successfully complete the challenges. Some of these are items to deploy to your development machine. Some are decisions you should discuss and define as a team, like the language to use for development.

1. **Azure Account**  
   You must have an Azure account to complete the hackathon. Either use your existing subscription or setup a free trial to complete today’s challenges.
2. **Optional: Setting up your Data Science Virtual Machine**

Access the [Azure portal](https://portal.azure.com). From the portal, [create a Resource Group for this lab](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-portal). Detailed information about the Data Science Virtual Machine can be [found online](https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/overview), but we will just go over what's needed for this workshop. We are creating a VM and not doing it locally to ensure that we are all working from the same environment. This will make troubleshooting much easier. In your Resource Group, deploy and connect to a "{CSP} Data Science Virtual Machine - Windows 2016", with a size of 2-4 vCPUs and 8-12 GB RAM, some examples include but are not limited to DS4\_V3, B4MS, DS3, DS3\_V2, etc. Select the location: West US 2. All other defaults are fine.

*> Note: Testing was completed on both West US 2 DS4\_V3 and Southeast Asia DS4\_V3.*

[Connect to your VM](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/connect-logon). Once you're connected, there are several things you need to do to set up the DSVM for the workshop:

* 1. In the Cortana search bar, type "git bash" and select "Git Bash Desktop App", or type "cmd" and select "Command Prompt". Next, type `cd c://` then enter, and `git clone https://github.com/Azure/LearnAI-Bootcamp.git` then enter. This copies down all of the files from the GitHub site to **C:\LearnAI-Bootcamp**.   
     *Validation step*: Go to **C:\LearnAI-Bootcamp** and confirm it exists.
  2. From File Explorer, open "ImageProcessing.sln" which is under **C:\LearnAI-Bootcamp\lab01.1-computer\_vision\resources\code\Starting-ImageProcessing**. It may take a while for Visual Studio to open for the first time, and you will have to log in to your account. The account you use to log in should be the same as your Azure subscription account.   
     *Note: If your company has two factor authentication, you may not be able to use your pin to log in. Use your password and mobile phone authentication to log in instead.*   
     *Validation step*: In the top right corner of Visual Studio, confirm that you see your name and account information.
  3. After Visual Studio loads and the solution is open, right-click on TestCLI and select "Set as StartUp Project."  
     *Validation step*: TestCLI should appear **bold** in the Solution Explorer   
     *Note: If you get a message that TestCLI is unable to load, right-click on TestCLI and select "Install Missing Features". This will prompt you to install* ***.NET Desktop Development****. Click* ***Install****, then* ***Install*** *again. You may get an error because Visual Studio needs to be closed to install updates. Close Visual Studio and then select* ***Retry****. It should only take 1-2 minutes to install. Reopen "ImageProcessing.sln", confirm that you are able to expand TestCLI and see its contents. Then, right-click on TestCLI and select "Set as StartUp Project".*
  4. Right-click on the solution in Solution Explorer and select "Build Solution".   
     *Validation step*: When you build the solution, the only errors you receive are related to `ImageProcessor` in `Program.cs`. You do not need to worry about yellow warning messages.

1. **Collecting the Keys**  
   Over the course of this lab, we will collect Cognitive Services keys and storage keys. You should save all of them in a text file so you can easily access them in future labs.
   1. Cognitive Services Keys  
      Computer Vision API:
   2. Storage Keys

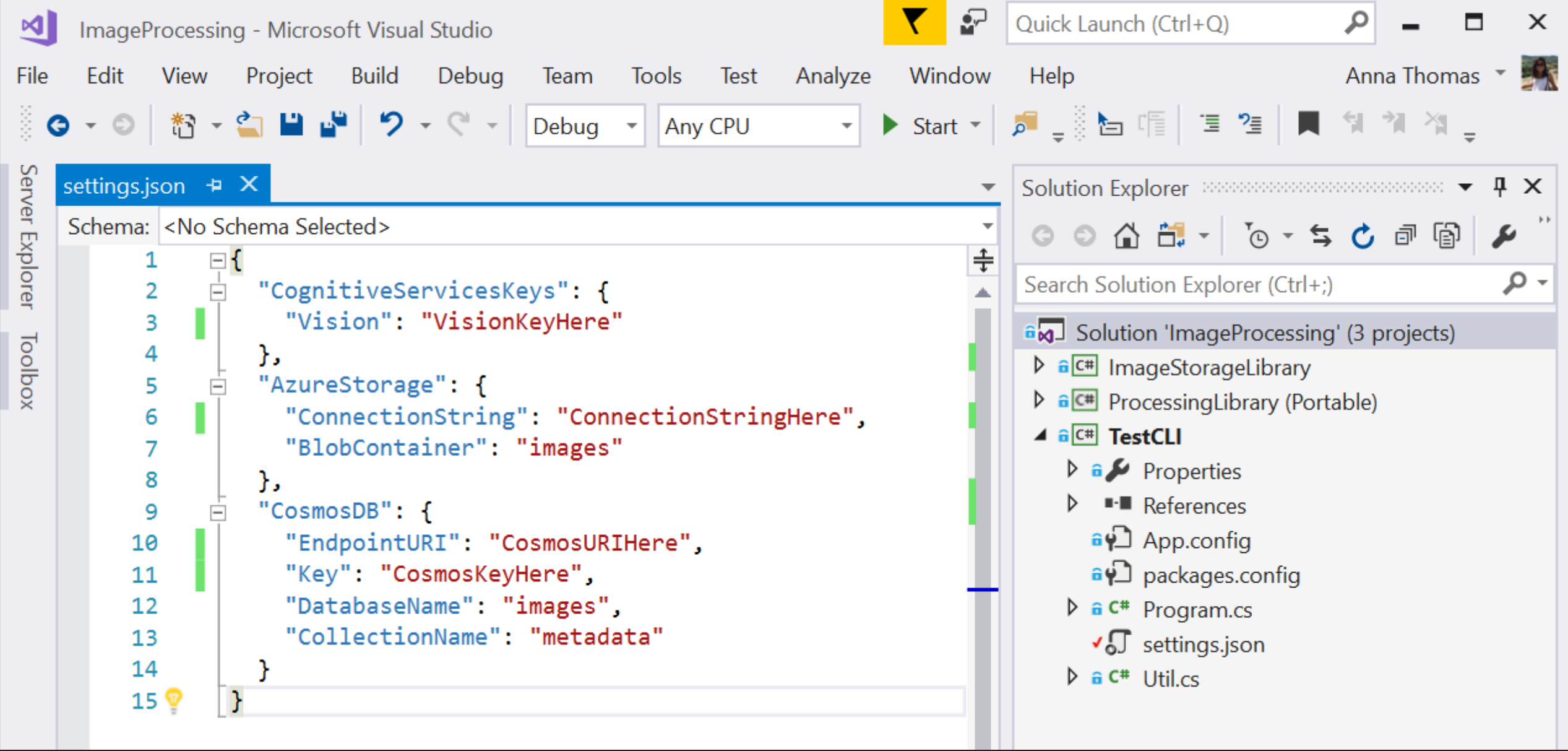
Azure Blob Storage Connection String:

Cosmos DB URI:

Cosmos DB key:

In addition, you will need to add the keys to the **settings.json** file which is under **C:\LearnAI-Bootcamp\lab01.1-computer\_vision\resources\code\Starting-ImageProcessing\TestCLI\settings.json**. You will have to replace `VisionKeyHere`, `ConnectionStringHere`, `CosmosURIHere`, and `CosmosKeyHere` with their corresponding keys that you collect in the next section.

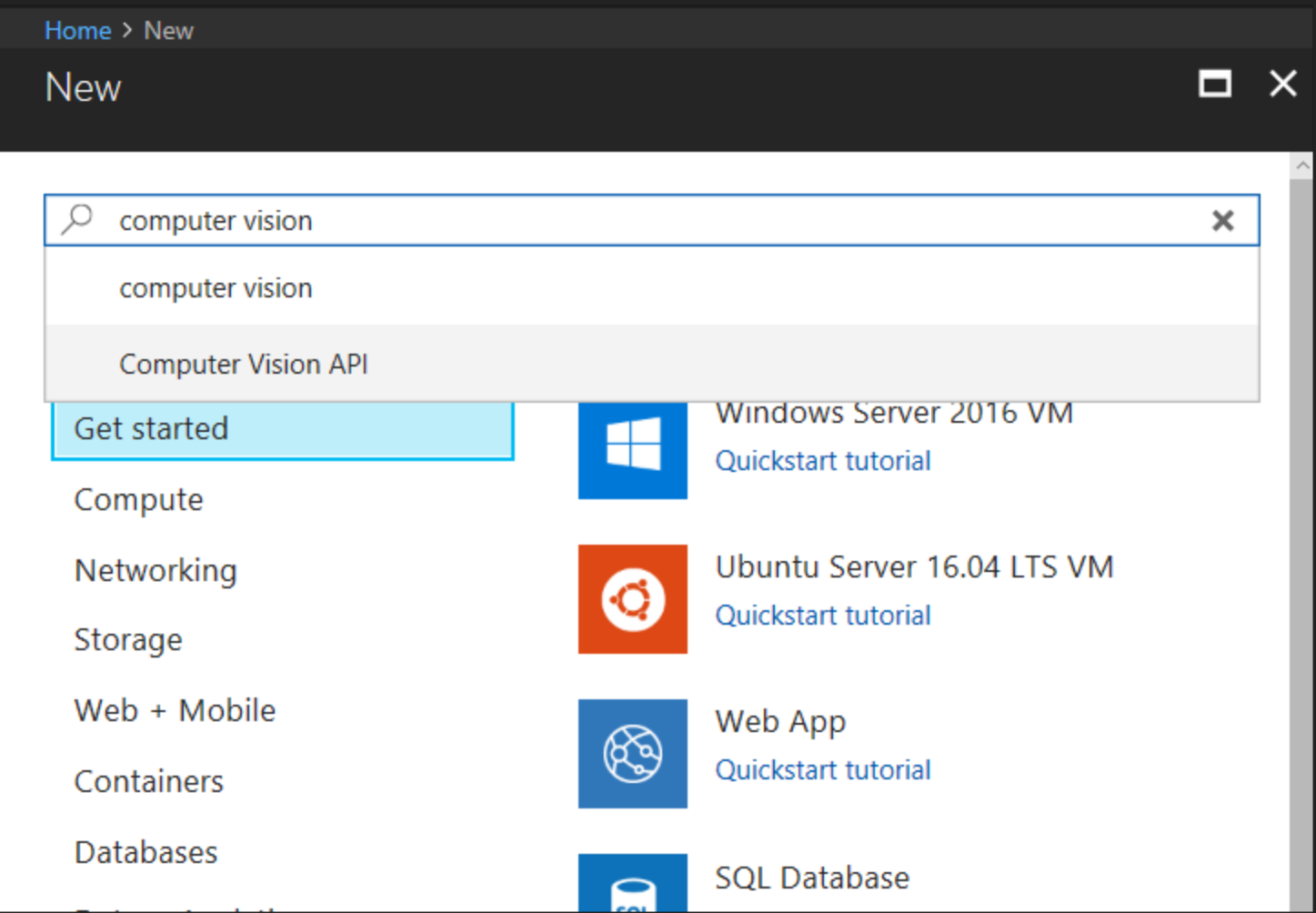
**Do not** change the blob container (`images`), the database name (`images`), or the collection name (`metadata`).



1. **Getting Cognitive Services API Keys**

Within the Azure Portal, we'll first create keys for the Cognitive Services we'll be using. We'll primarily be using the [Computer Vision](https://www.microsoft.com/cognitive-services/en-us/computer-vision-api) Cognitive Service, so let's create an API key for that first.

In the Portal, click the "+ New" button (when you hover over it, it will say **Create a resource**) and then enter **computer vision** in the search box and choose **Computer Vision API**:

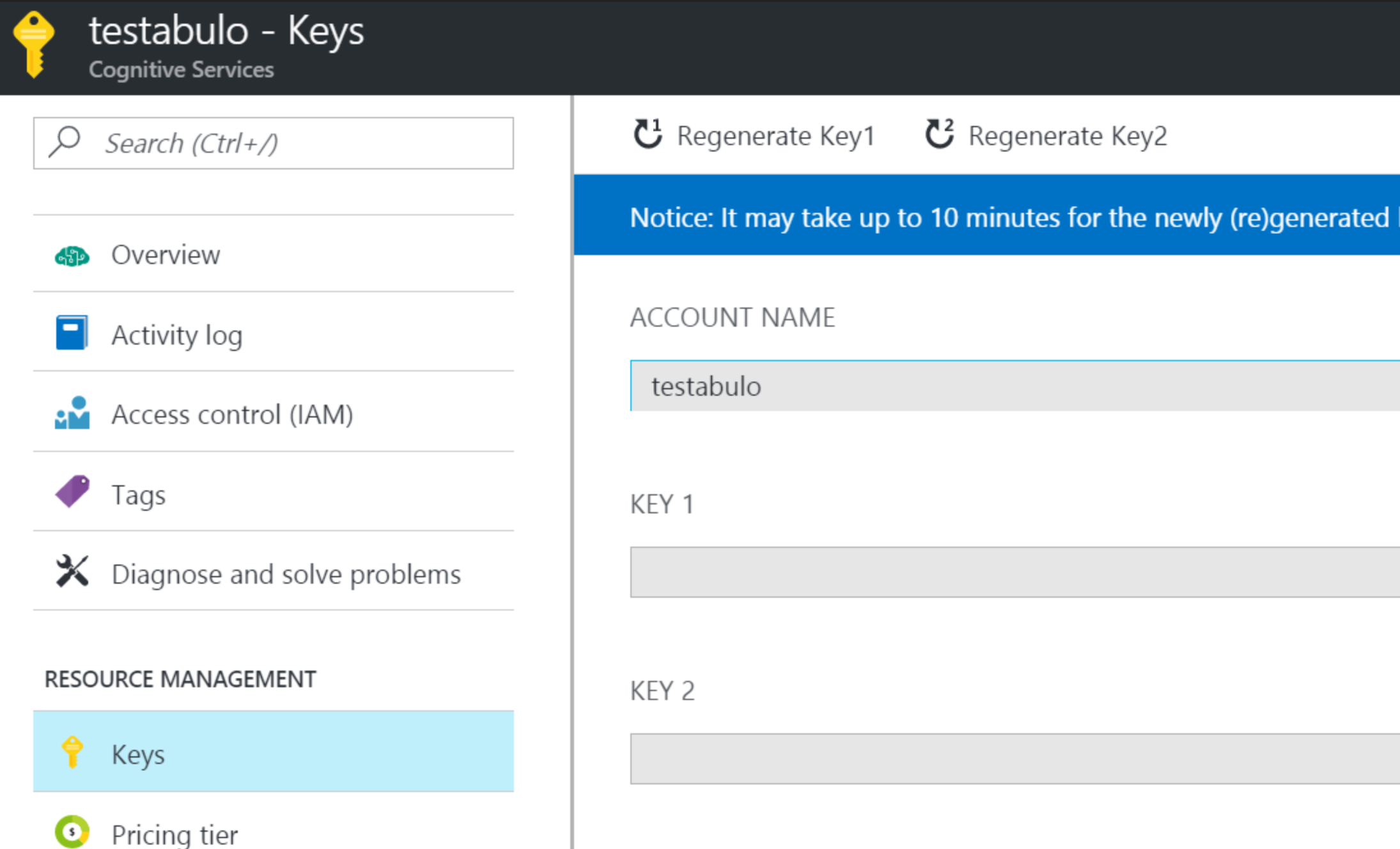


This will lead you to fill out a few details for the API endpoint you'll be creating, choosing the API you're interested in and where you'd like your endpoint to reside (**put in the West US region or it will not work**), as well as what pricing plan you'd like. We'll be using **S1** so that we have the throughput we need for the tutorial. Use the same Resource Group that you used to create your DSVM. We'll also use this resource group for Blob Storage and Cosmos DB. **Pin to dashboard** so that you can easily find it. Since the Computer Vision API stores images internally at Microsoft (in a secure fashion), to help improve future Cognitive Services Vision offerings, you'll need to check the box that states you're ok with this before you can create the resource.

**Double check that you put your Computer Vision service in West US**

The code in the following labs has been set up to use West US for calling the Computer Vision API. In your future endeavors, you can learn how to call other regions [here](https://docs.microsoft.com/en-us/azure/cognitive-services/Computer-vision/Vision-API-How-to-Topics/HowToSubscribe).

1. **Modifying `settings.json`, part one**Once you have created your new API subscription, you can grab the keys from the appropriate section of the blade and add them to your TestCLI's `settings.json` file.



Note: there are two keys for each of the Cognitive Services APIs you will create. Either one will work. You can read more about multiple keys [here](https://blogs.msdn.microsoft.com/mast/2013/11/06/why-does-an-azure-storage-account-have-two-access-keys/).

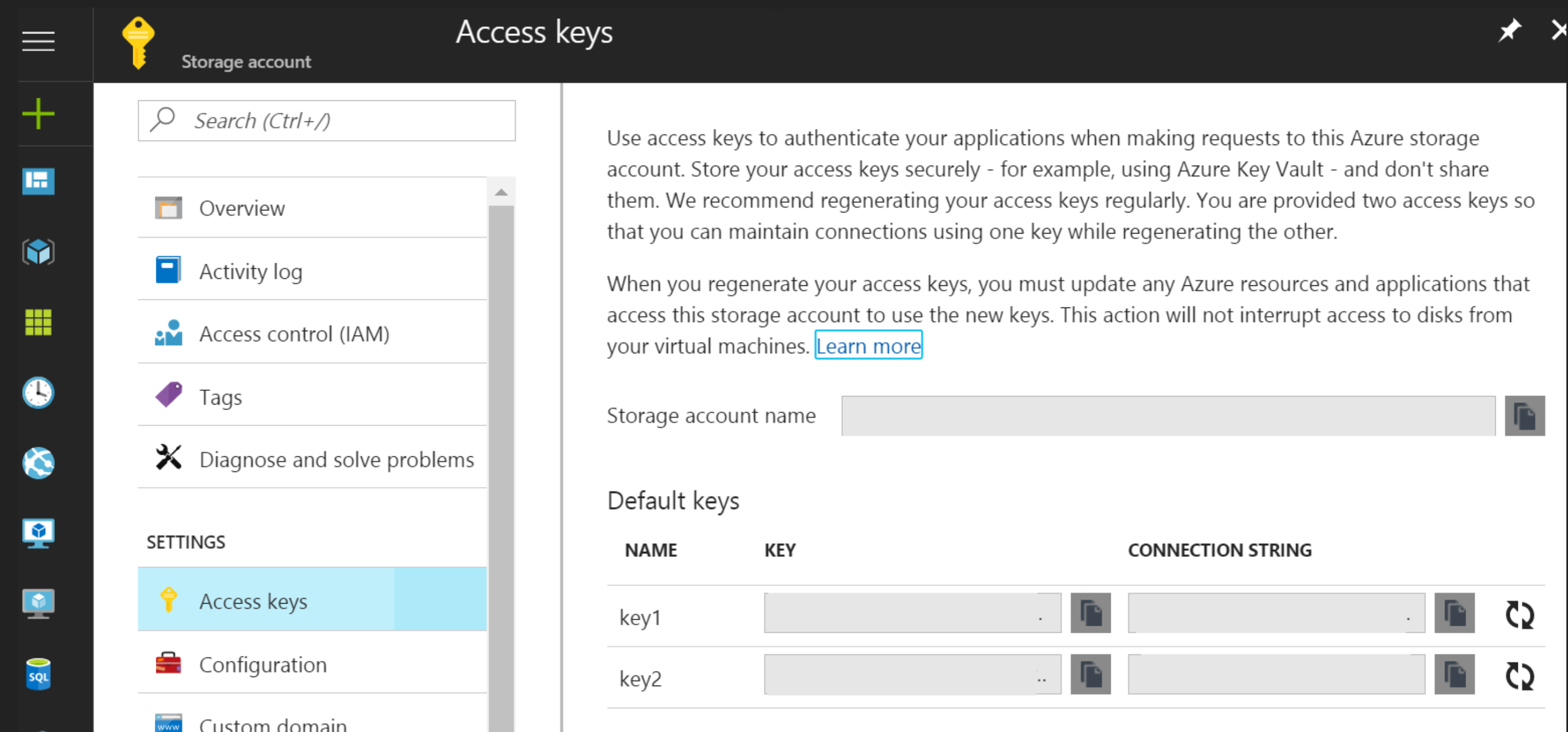
1. **Setting up Storage**  
   We'll be using two different stores in Azure for this project - one for storing the raw images, and the other for storing the results of our Cognitive Service calls. Azure Blob Storage is made fostoring large amounts of data in a format that looks similar to a file-system, and it is a great choice for storing data like images. Azure Cosmos DB is our resilient NoSQL PaaS solution and is incredibly useful for storing loosely structured data like we have with our image metadata results. There are other possible choices (Azure Table Storage, SQL Server), but Cosmos DB gives us the flexibility to evolve our schema freely (like adding data for new services), query it easily, and integrate quickly into Azure Search.
   1. **Azure Blob Storage**  
      Detailed "Getting Started" instructions can be [found online](https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-blobs), but let's just go over what you need for this lab.

In the Portal, click the **"+ New"** button (when you hover over it, it will say **Create a resource**) and then enter **storage** in the search box and choose **Storage account**. Select **create**.

Once you click it, you'll be presented with some fields to fill out.

* Choose your storage account name (lowercase letters and numbers)
* Set **Account kind** to “Blob storage”
* Set **Replication** to “Locally-Redundant storage (LRS)” (this is just to save money)
* Use the same **Resource Group** as above
* Set **Location** to the region that is closest to you from the following list: East US, West US, Southeast Asia, West Europe. (The list of Azure services that are available in each region is at <https://azure.microsoft.com/en-us/regions/services>). **Pin to dashboard**, so that you can easily find it.
* All other defaults are fine
  1. **Modifying settings.json, part two**

Now that you have an Azure Storage account, grab the **Connection String** and add it to your **TestCLI**'s settings.json.

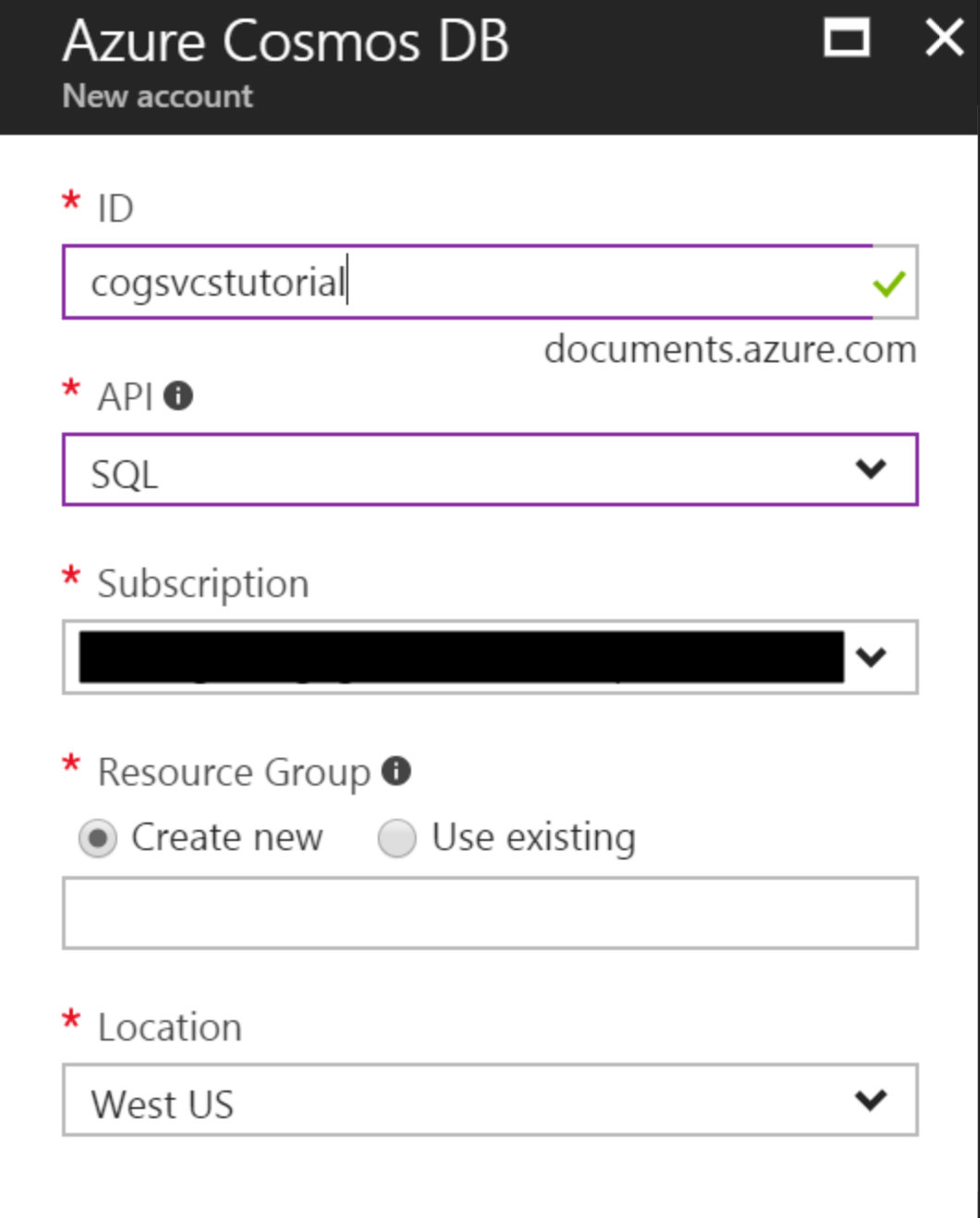


* 1. **Cosmos DB**

Detailed "Getting Started" instructions can be [found online](https://docs.microsoft.com/en-us/azure/cosmos-db/documentdb-get-started), but we'll walk through what you need for this lab.

In the Portal, click the "**+ New**" button (when you hover over it, it will say **Create a resource**) and then enter **cosmos db** in the search box and choose **Azure Cosmos DB** and click **Create**.

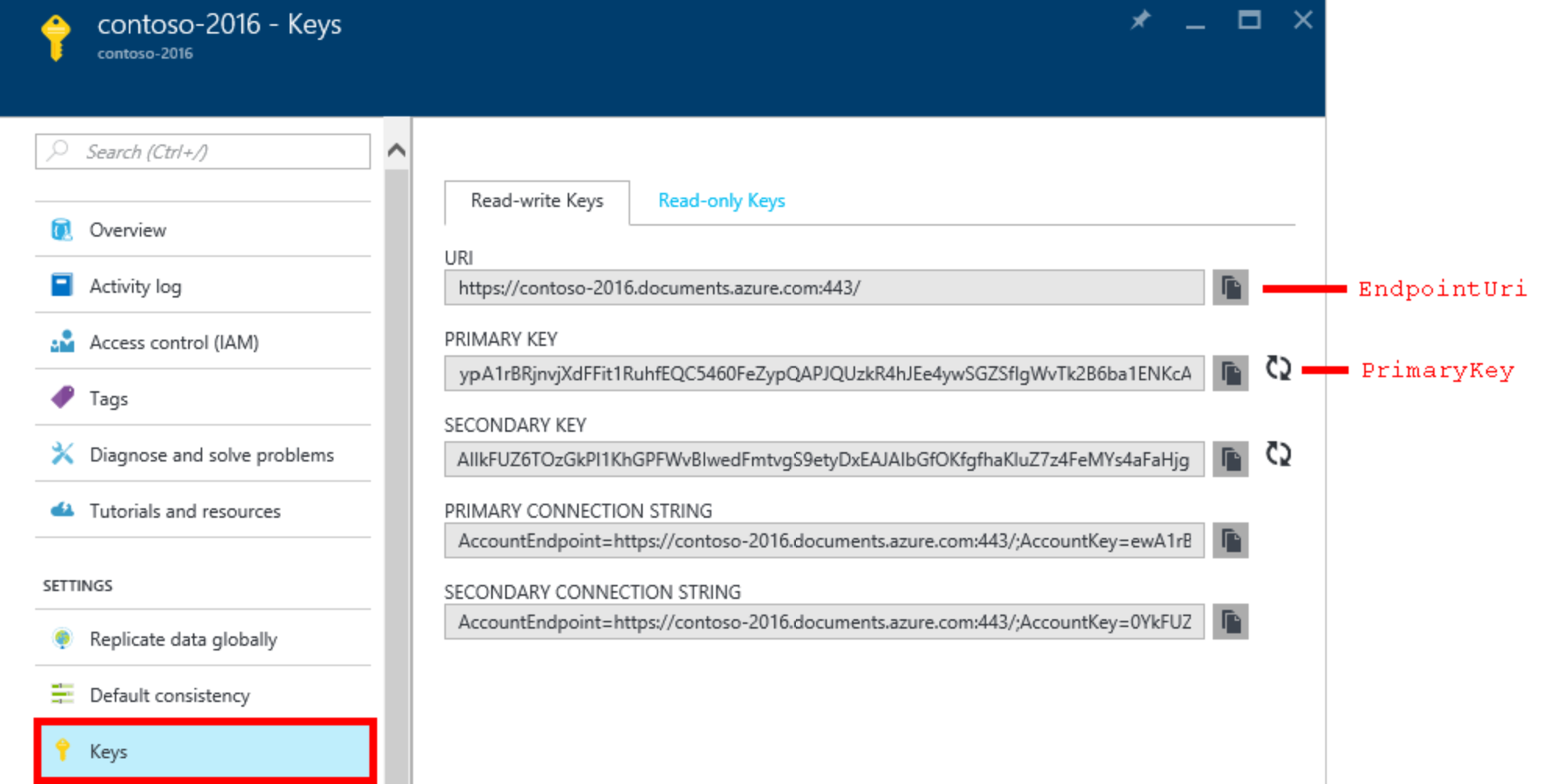
Once you click this, you'll have to fill out a few fields as you see fit. Set \_Location\_ to the region that is closest to you from the following list: East US, West US, Southeast Asia, West Europe.



In our case, select the ID you'd like, subject to the constraints that it needs to be lowercase letters, numbers, or dashes. We will be using the SQL API so we can create a document database that is queryable using SQL syntax, so select `SQL` as the API. Let's use the same Resource Group as we used for our previous steps, and the same location, select \_Pin to dashboard\_ to make sure we keep track of it and it's easy to get back to, and hit Create.

* 1. **Modifying settings.json, part three**

Once creation is complete, open the panel for your new database and select the \_Keys\_ sub-panel.



You'll need the **URI** and the **PRIMARY KEY** for your **TestCLI's** **settings.json** file, so copy those into there and you're now ready to store images and data into the cloud.

*Note: Be sure to turn off your DSVM* ***from the portal*** *after you have completed the Setup lab. When the workshop begins, you will need to start your DSVM from the portal to begin the labs. We recommend turning off your DSVM at the end of each day, and deleting all of the resources you create at the end of the workshop. Alternatively, you can* [*set up auto-shutdown for your DSVM*](https://blogs.msdn.microsoft.com/devtestlab/2018/01/02/set-auto-shutdown-for-virtual-machines-in-azure/)*. Be sure to set the correct time zone.*

You have completed the prerequisites.

Configuring the following for Custom Vision:

* Training API Key: The training API key allows you to create, manage and train Custom Vision project programmatically.
* You can obtain a key by creating a new project at <https://customvision.ai> and then clicking on the “setting” gear in the top right.

## Working together

* Will your team work together or individually?
* Git for Windows – allows for working with Git repos locally on your machine as well as in VSTS. <https://gitscm.com/downloads>
* Determine Azure Subscription you will use for deployment

## Resources

There are several folders in the Team site, including resources folders for each challenge.

# Challenges

## Part 1: Cognitive Services

This challenge involves building an end-to-end application that allows you to pull in your own pictures, use Cognitive Services to obtain a caption and some tags about the images, and then store that information in Cosmos DB. Build on your work by using the NoSQL store (Cosmos DB) to populate an Azure Search index, and then build a Bot Framework bot using LUIS to allow easy, targeted querying.

**Challenge Summary**

Your team is preparing a proof of concept, and your group has been tasked to suggest answers to the following questions:

1. What might your main intents in LUIS be?

2. Are there any additional Cognitive Services that you think could be used to help bring value to the business?

3. What are some potential ideas for how Custom Vision could be used to bring value?

Document your suggestions on the whiteboards/flipcharts provided. You will present your results.

[Business Case I - Cognitive Services](about:./lab01.6-bootcamp_case_1/Case_Part1.md) – Broken link? Where?

* Setup assistance & Introduction and Context for Cognitive Services – See 1\_Setup.md in the AI Team site > Files > challenge1.1-computer\_vision
* Challenge 1.1: Simplifying Cognitive Services App Development using Portable Class Libraries
* Challenge 1.2: Creating an Image Classification Application using the Custom Vision Service I
* Challenge 1.3: Creating an Image Classification Application using the Custom Vision Service II
* Challenge 1.4: Creating an Image Classification Application using the Custom Vision Service III
* Challenge 1.5: Developing Intelligent Applications with LUIS

## Part 2: Bots

You are working with Contoso LLC, which sells bicycles and bicycle equipment to

its customers. Contoso currently processes new product orders and queries

through human operators, and are starting to devise a plan to implement your

proposed solution using bots. The solution will provide an automated approach

that allows Contoso to seamlessly scale up to handle a large call volumes while

maintaining zero wait times and freeing up staff to manage other tasks.

**Challenge Summary**

Your group has been tasked to suggest answers to the following questions:

1. Which features from Azure Search should you take advantage of?

2. How can bot logging be used to benefit Contoso?

3. What are some tasks you'll have to complete to create an efficient and functional calling bot?

Work in team of 4 or 5 as assigned by the instructor to discuss the options that

are available. This will be time limited between 20 - 30 mins.

[Business Case II - Bots](about:./lab02.5-bootcamp_case_2/Case_Part2.md) – Broken link? Where?

* Introduction and Context for Bots
* Challenge 2.1: Developing Intelligent Applications with Azure Search
* Challenge 2.2: Building Intelligent Bots
* Challenge 2.3: Log Chat Conversations in your Bot
* Challenge 2.4: Testing your Bot
* Q&A and Feedback