**Lab14 – Understanding Blob Storage - Azure**

**Blob storage**

Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data. Unstructured data is data that does not adhere to a particular data model or definition, such as text or binary data.

**About Blob storage**

Blob storage is designed for:

* Serving images or documents directly to a browser.
* Storing files for distributed access.
* Streaming video and audio.
* Writing to log files.
* Storing data for backup and restore, disaster recovery, and archiving.
* Storing data for analysis by an on-premises or Azure-hosted service.

Users or client applications can access objects in Blob storage via HTTP/HTTPS, from anywhere in the world. Objects in Blob storage are accessible via the [Azure Storage REST API](https://docs.microsoft.com/rest/api/storageservices/blob-service-rest-api), [Azure PowerShell](https://docs.microsoft.com/powershell/module/azure.storage), [Azure CLI](https://docs.microsoft.com/cli/azure/storage), or an Azure Storage client library. Client libraries are available for a variety of languages, including [.NET](https://docs.microsoft.com/dotnet/api/overview/azure/storage/client), [Java](https://docs.microsoft.com/java/api/overview/azure/storage/client), [Node.js](http://azure.github.io/azure-storage-node), [Python](https://docs.microsoft.com/python/azure/), [Go](https://github.com/azure/azure-storage-blob-go/), [PHP](http://azure.github.io/azure-storage-php/), and [Ruby](http://azure.github.io/azure-storage-ruby).

**About Azure Data Lake Storage Gen2**

Blob storage supports Azure Data Lake Storage Gen2, Microsoft's enterprise big data analytics solution for the cloud. Azure Data Lake Storage Gen2 offers a hierarchical file system as well as the advantages of Blob storage, including low-cost, tiered storage; high availability; strong consistency; and disaster recovery capabilities.

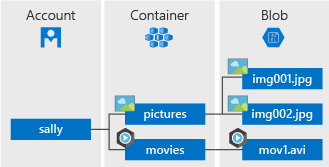
For more information about Data Lake Storage Gen2, see [Introduction to Azure Data Lake Storage Gen2 Preview](https://docs.microsoft.com/en-us/azure/storage/data-lake-storage/introduction).

**Blob storage resources**

Blob storage offers three types of resources:

* The **storage account**.
* A **container** in the storage account
* A **blob** in a container

The following diagram shows the relationship between these resources.



**Storage accounts**

A storage account provides a unique namespace in Azure for your data. Every object that you store in Azure Storage has an address that includes your unique account name. The combination of the account name and the Azure Storage service endpoint forms the endpoints for your storage account.

**Containers**

A container organizes a set of blobs, similar to a directory in a file system. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.

The container name must be lowercase. For more information about naming containers, see [**Naming and Referencing Containers, Blobs, and Metadata**](https://docs.microsoft.com/rest/api/storageservices/Naming-and-Referencing-Containers--Blobs--and-Metadata).

**Blobs**

Azure Storage supports three types of blobs:

* **Block blobs** store text and binary data, up to about 4.7 TB. Block blobs are made up of blocks of data that can be managed individually.
* **Append blobs** are made up of blocks like block blobs, but are optimized for append operations. Append blobs are ideal for scenarios such as logging data from virtual machines.
* **Page blobs** store random access files up to 8 TB in size. Page blobs store the virtual hard drive (VHD) files serve as disks for Azure virtual machines. Fore more information about page blobs, see (../articles/storage/blobs/storage-blob-pageblob-overview.md)

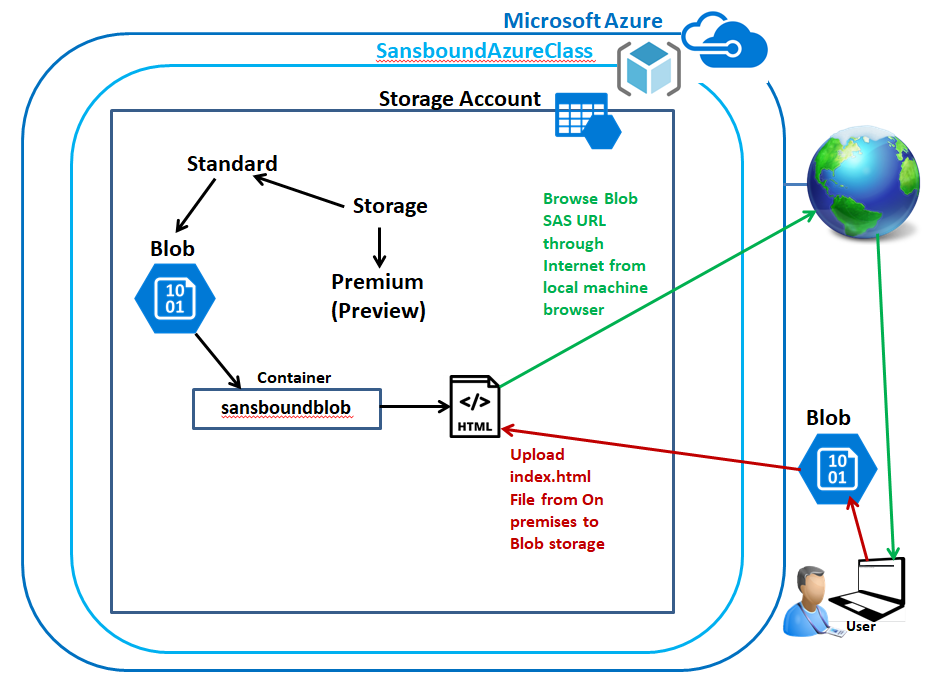
For more information about the different types of blobs, see [Understanding Block Blobs, Append Blobs, and Page Blobs](https://docs.microsoft.com/rest/api/storageservices/understanding-block-blobs--append-blobs--and-page-blobs).

**Move data to Blob storage**

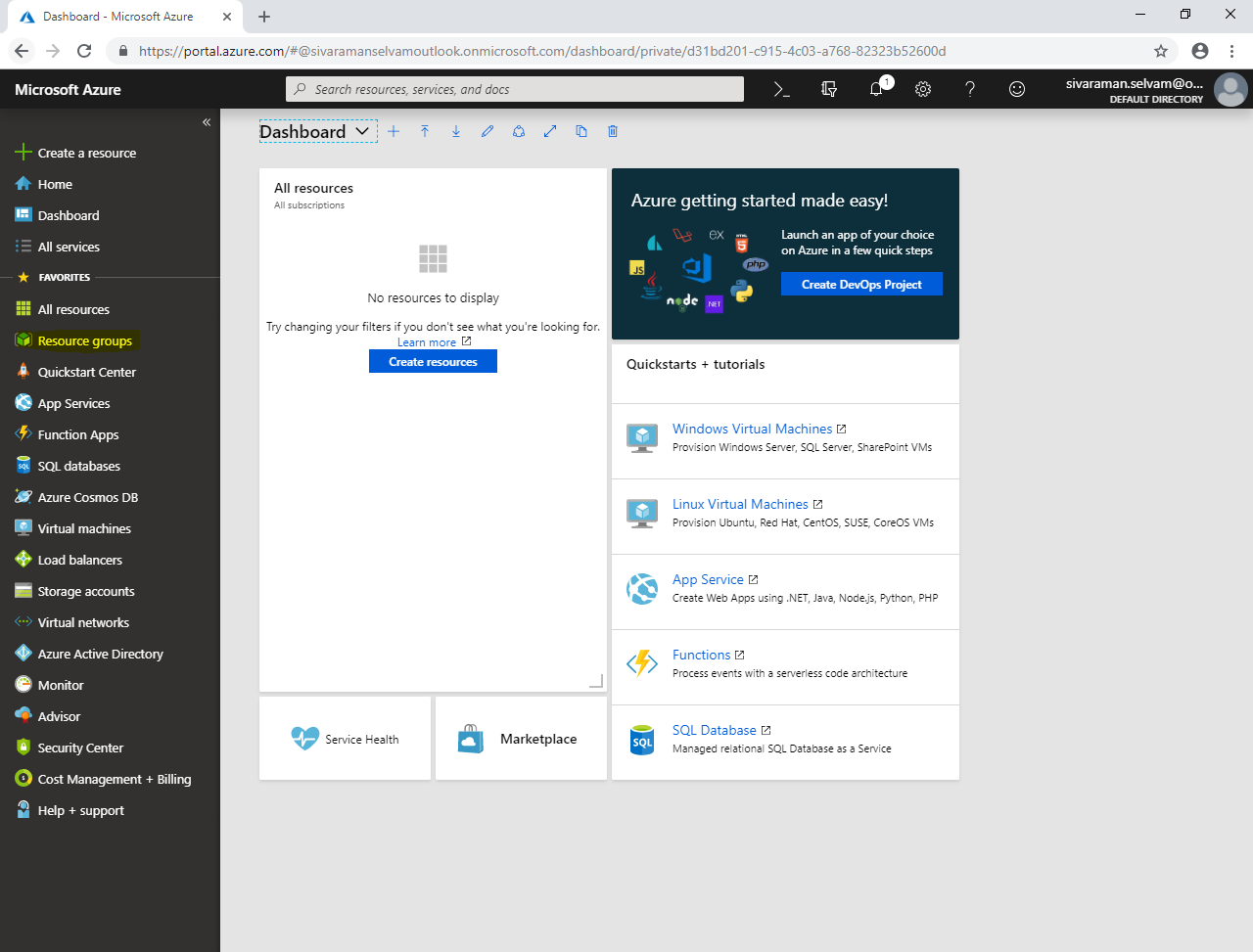
A number of solutions exist for migrating existing data to Blob storage:

* **AzCopy** is an easy-to-use command-line tool for Windows and Linux that copies data to and from Blob storage, across containers, or across storage accounts. For more information about AzCopy, see [Transfer data with the AzCopy v10 (Preview)](https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10).
* The **Azure Storage Data Movement library** is a .NET library for moving data between Azure Storage services. The AzCopy utility is built with the Data Movement library. For more information, see the [reference documentation](https://docs.microsoft.com/dotnet/api/microsoft.windowsazure.storage.datamovement) for the Data Movement library.
* **Azure Data Factory** supports copying data to and from Blob storage by using the account key, shared access signature, service principal, or managed identities for Azure resources authentications. For more information, see [Copy data to or from Azure Blob storage by using Azure Data Factory](https://docs.microsoft.com/azure/data-factory/connector-azure-blob-storage?toc=%2fazure%2fstorage%2fblobs%2ftoc.json).
* **Blobfuse** is a virtual file system driver for Azure Blob storage. You can use blobfuse to access your existing block blob data in your Storage account through the Linux file system. For more information, see [How to mount Blob storage as a file system with blobfuse](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-how-to-mount-container-linux).
* **Azure Data Box Disk** is a service for transferring on-premises data to Blob storage when large datasets or network constraints make uploading data over the wire unrealistic. You can use [Azure Data Box Disk](https://docs.microsoft.com/en-us/azure/databox/data-box-disk-overview) to request solid-state disks (SSDs) from Microsoft. You can then copy your data to those disks and ship them back to Microsoft to be uploaded into Blob storage.
* The **Azure Import/Export service** provides a way to export large amounts of data from your storage account to hard drives that you provide and that Microsoft then ships back to you with your data. For more information, see [Use the Microsoft Azure Import/Export service to transfer data to Blob storage](https://docs.microsoft.com/en-us/azure/storage/common/storage-import-export-service).

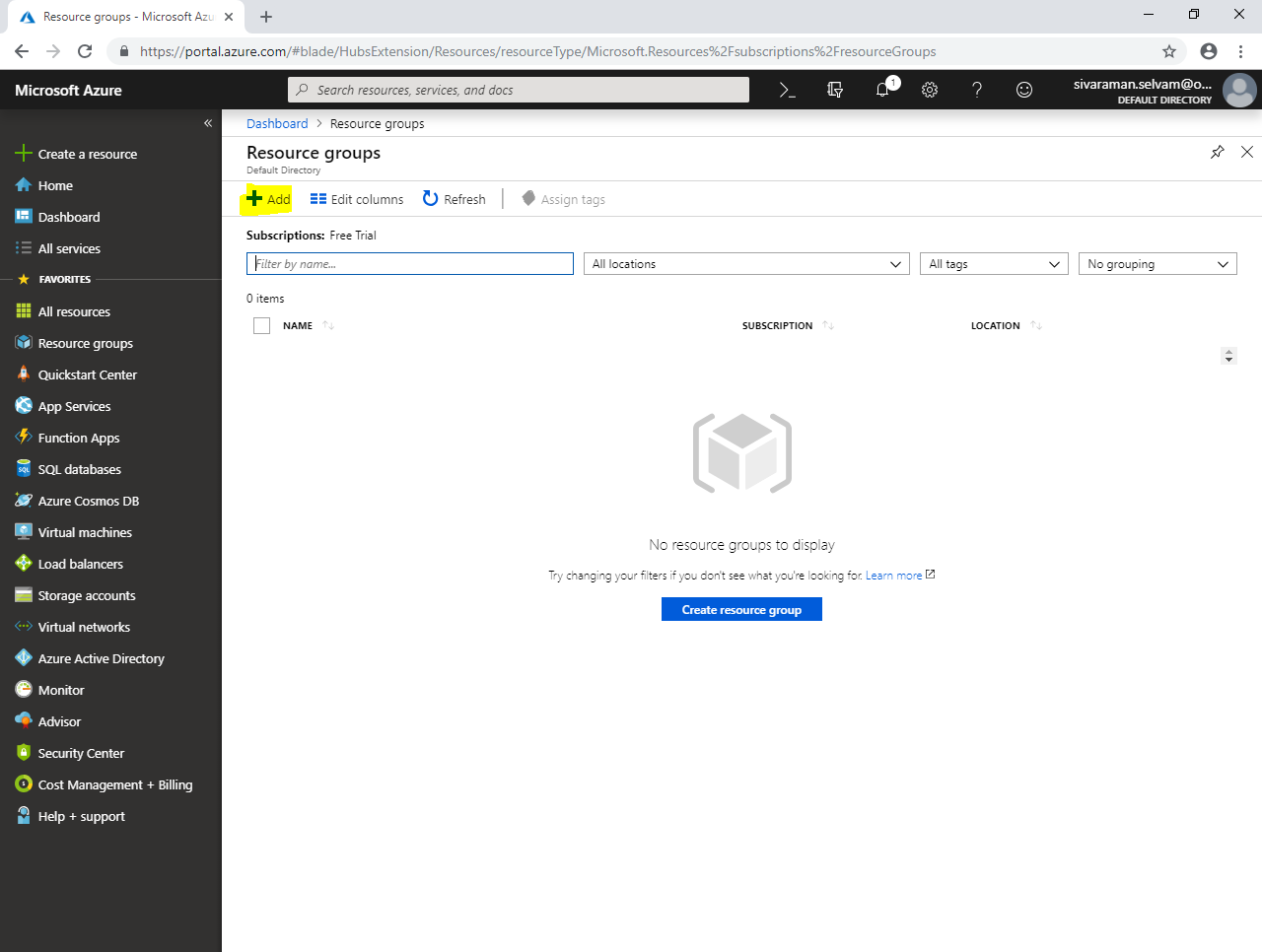
**Topology**



In Azure portal, click **“Resource groups”**.



Click **“Add”**.

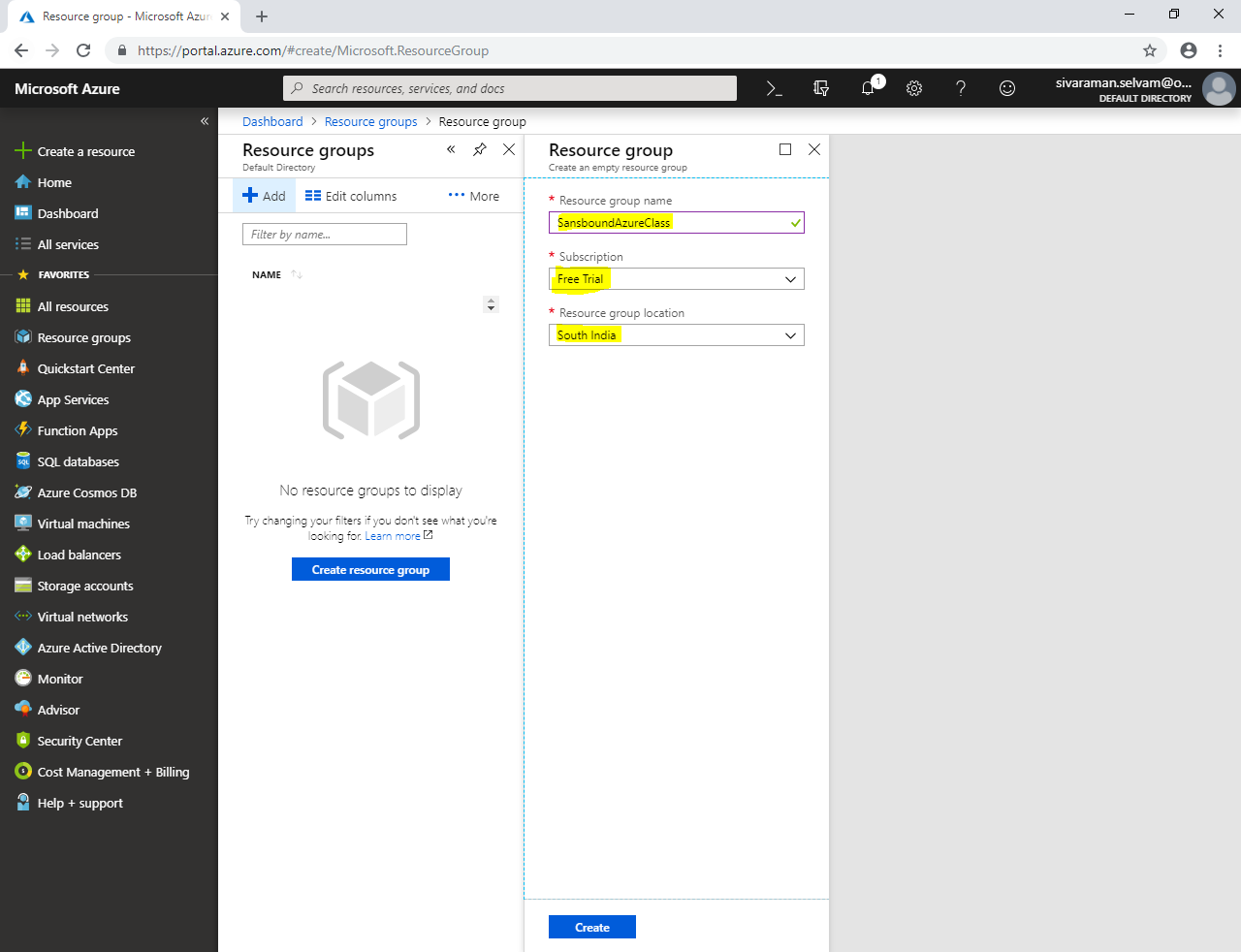


While create **“Resource group”**.

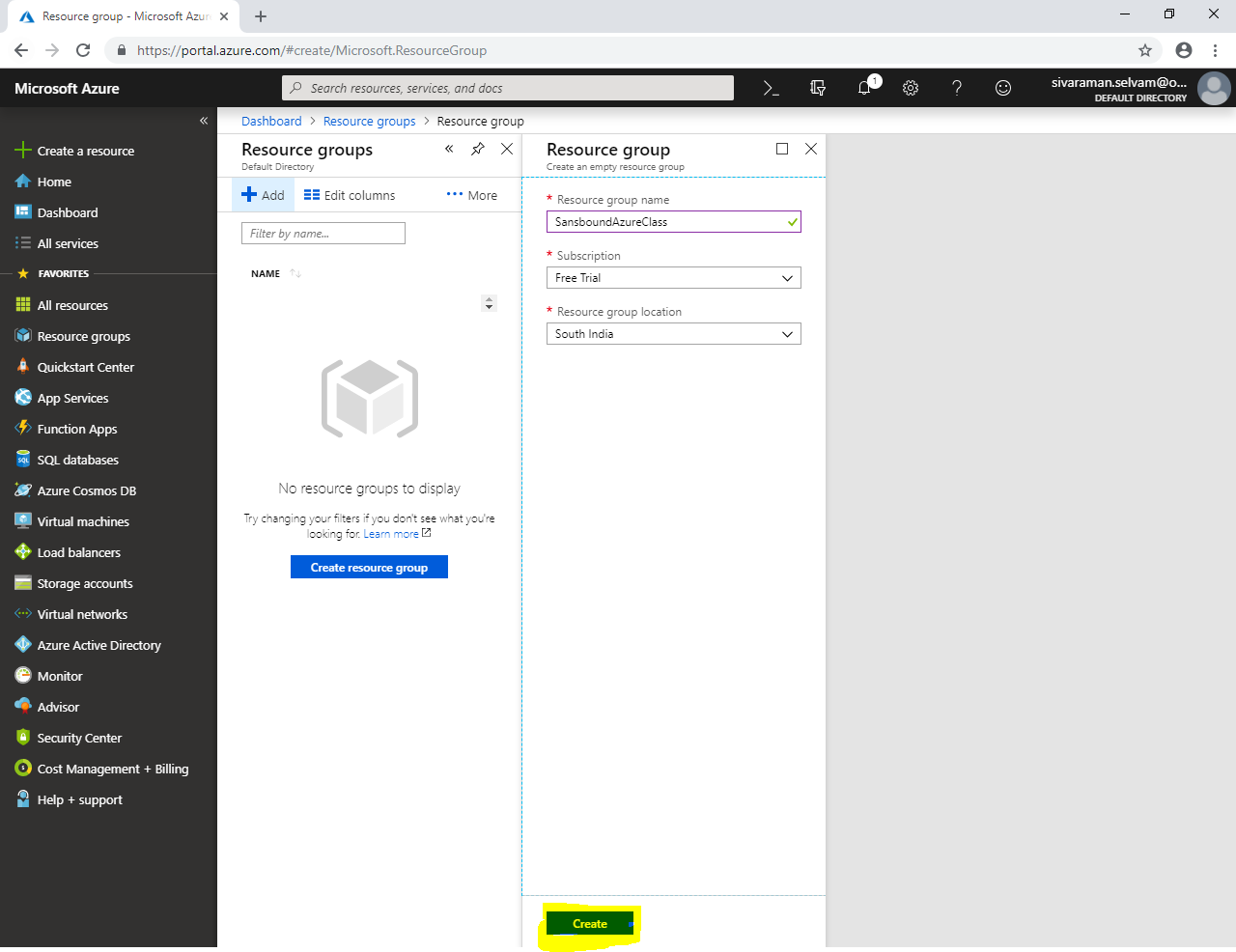
Type **“Resource group name”** as **“SansboundAzureClass”**.

Select **“Subscription”** as **“Free Trial”**.

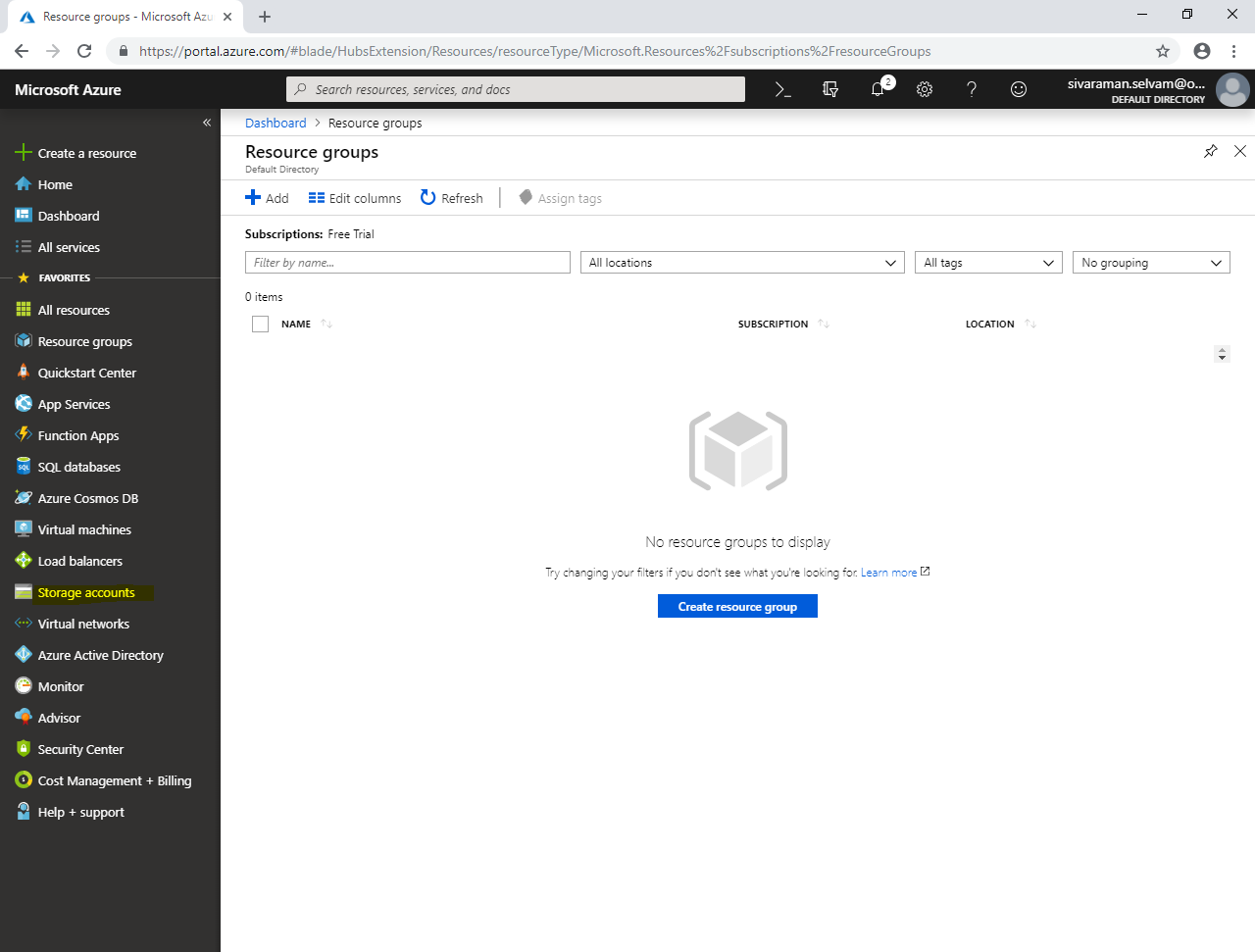
Select **“Resource group location”** as **“South India”**.



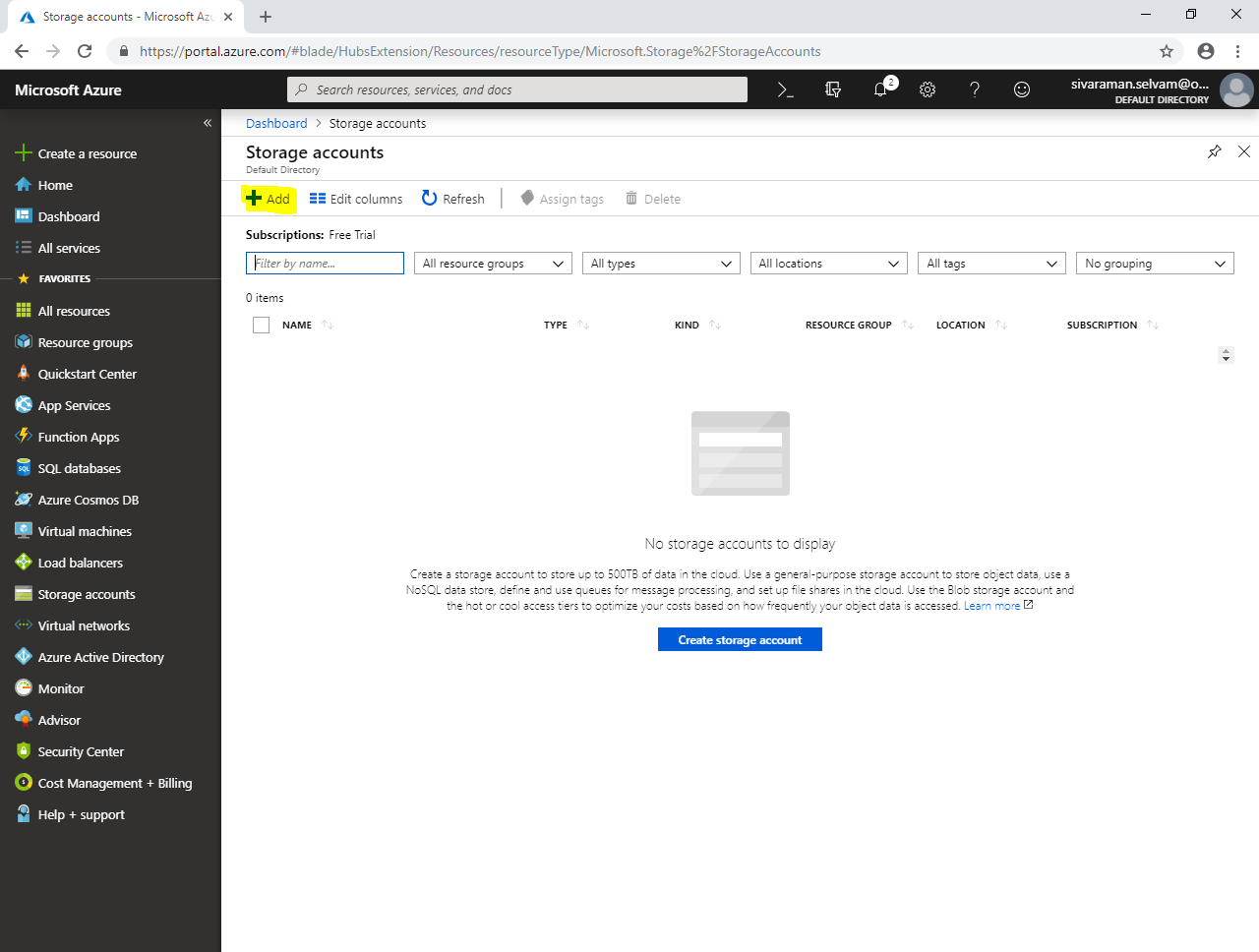
Click **“Create”**.



Click **“Storage accounts”** in left side panel.



Click **“Add”**.



While create storage account,

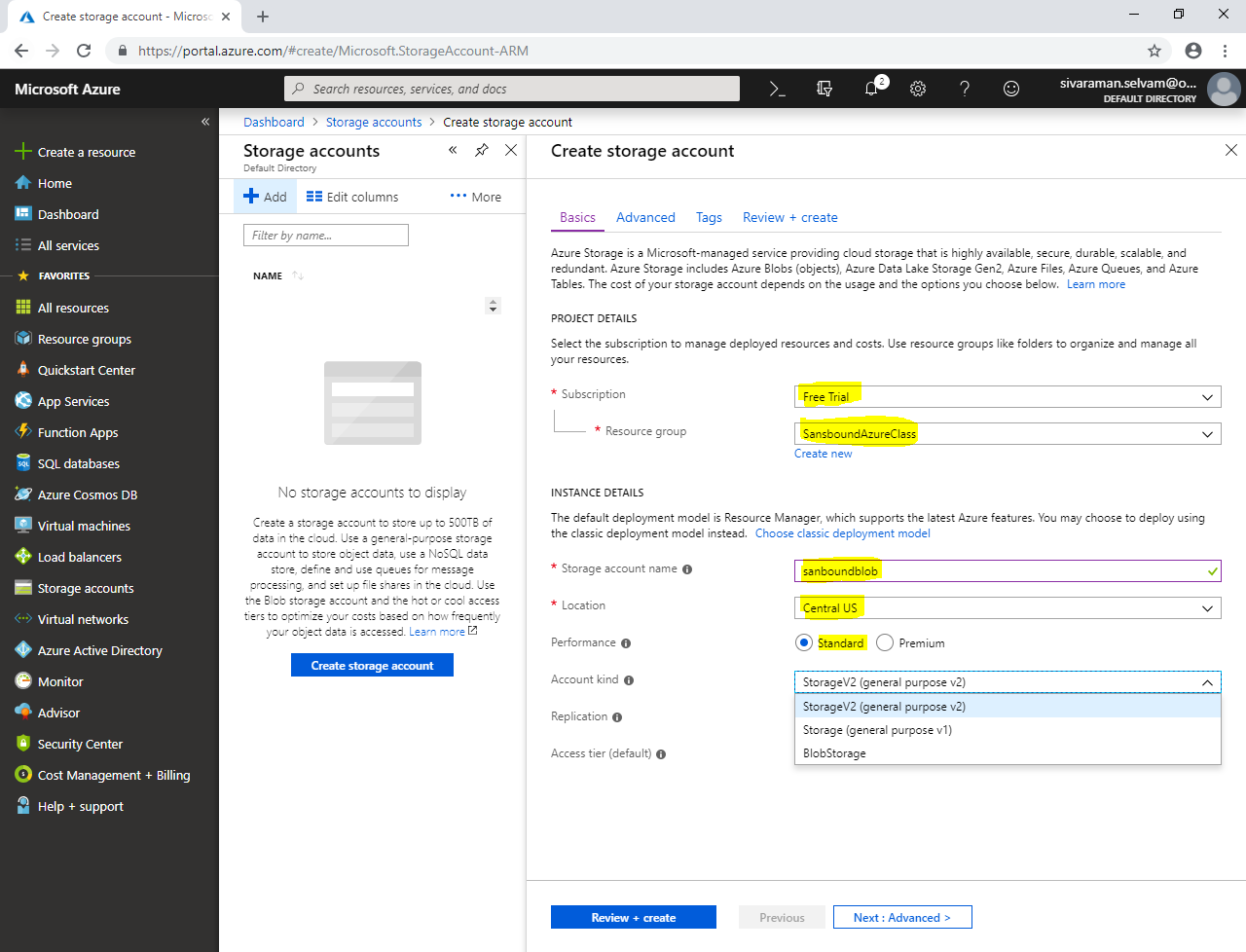
Select **“Subscription”** as **“Free Trial”**.

Select **“Resource group”** as **“SansboundAzureClass”**.

Type **“Storage account name”** as **“sansboundblob”**.

Select **“Location”** as **“Central US”**.

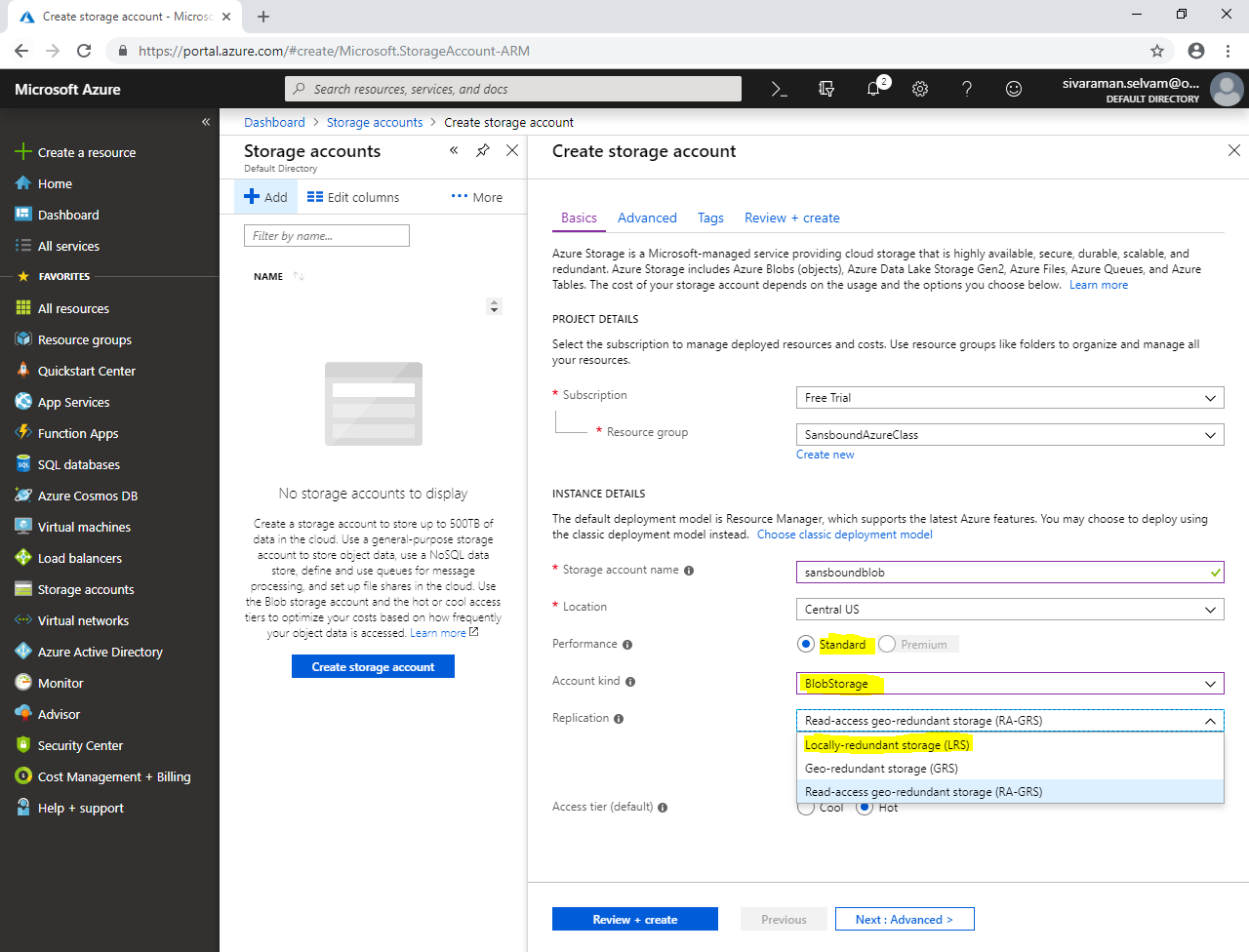
Set Performance as **“Standard”** (You can select “Premium” also, but it’s in preview state, that means they are introducing new feature / in under testing not fully completed).



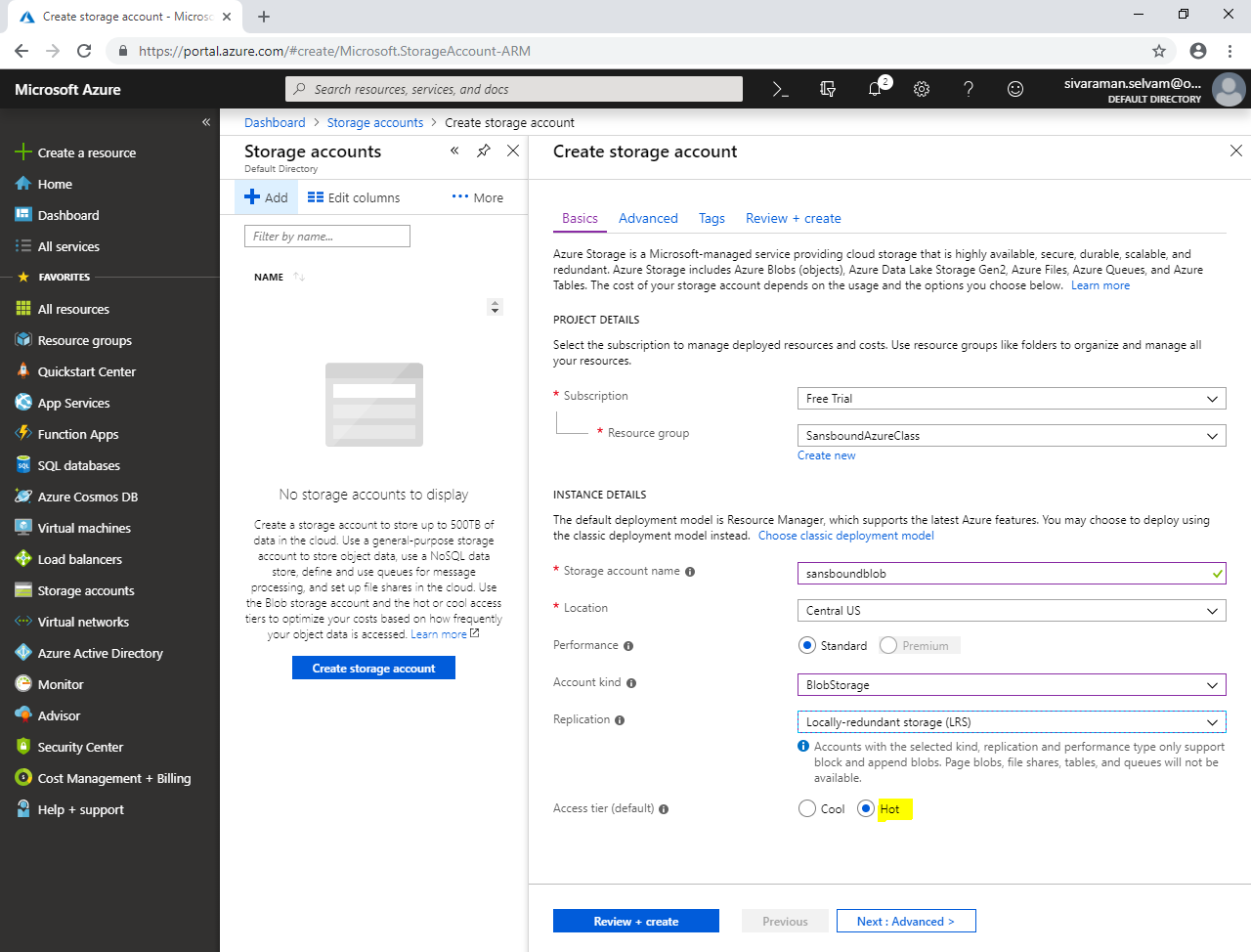
In **“Performance”** click **“Standard”**.

Select **“Account kind”** as **“Blob Storage”**.

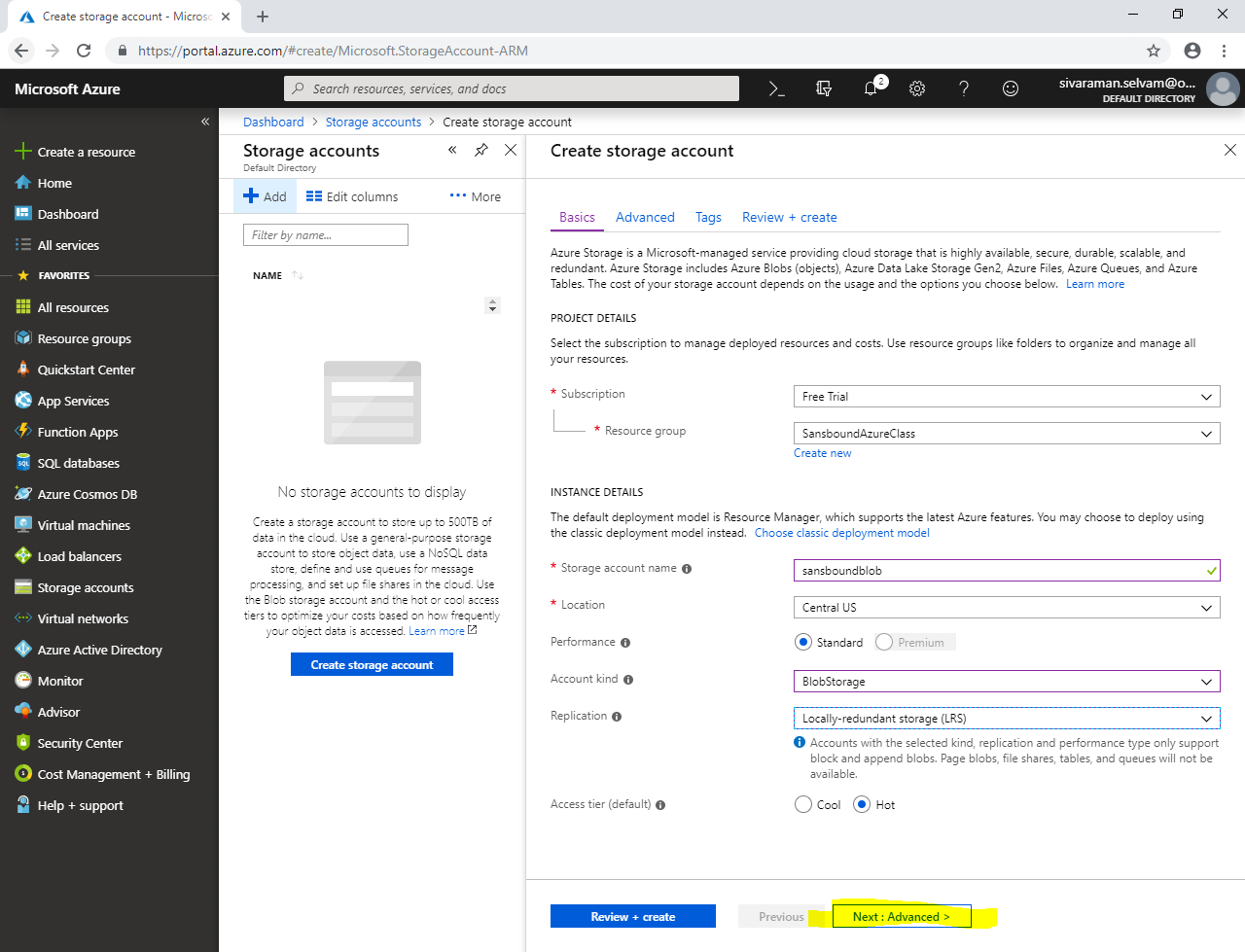
Select **“Replication”** as **“Locally-redundant storage”** (We will discuss LRS feature later briefly).



Set **“Access tier”** as **“Hot”**.

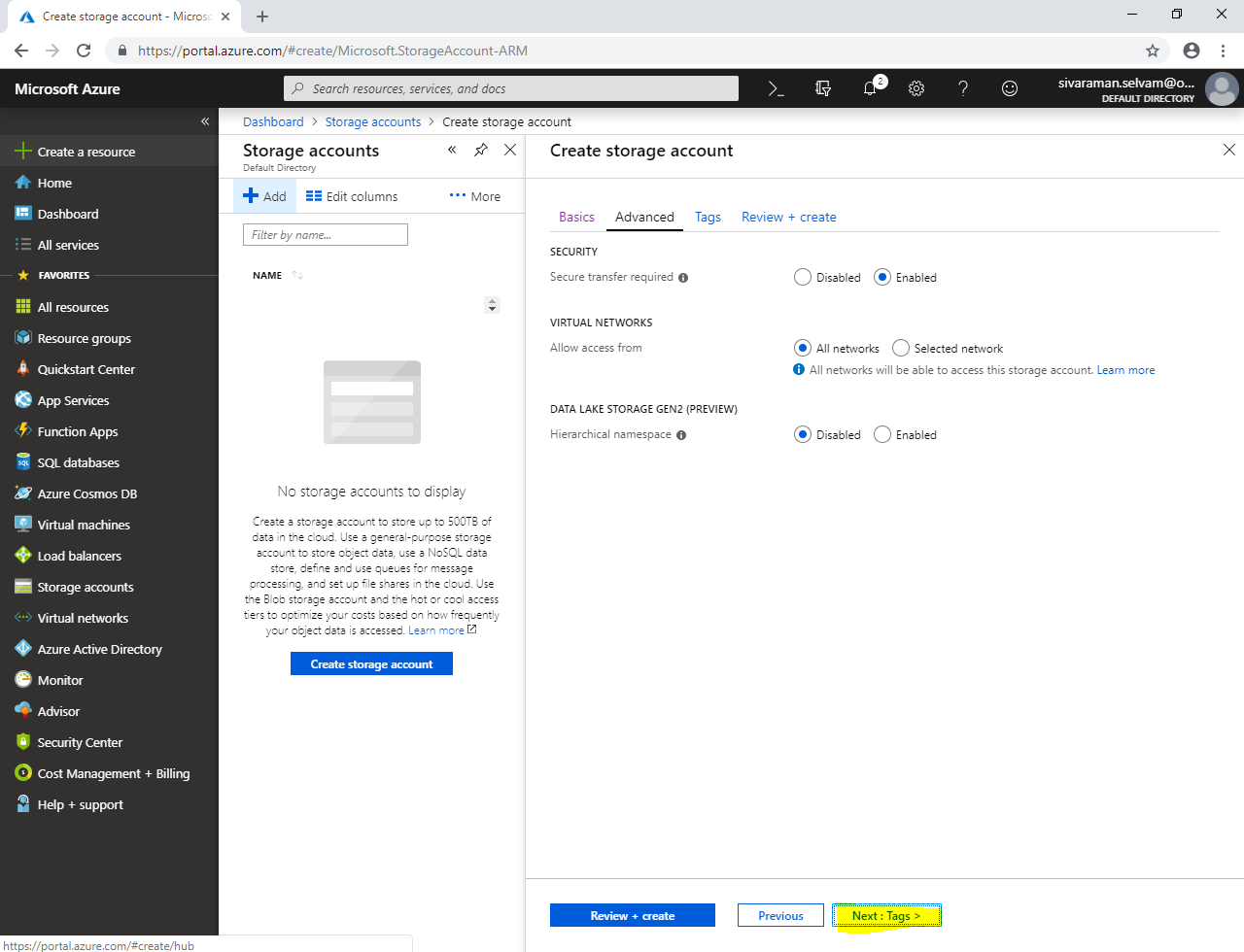


Click **“Next : Advanced >”**.



In **“Advanced”**.

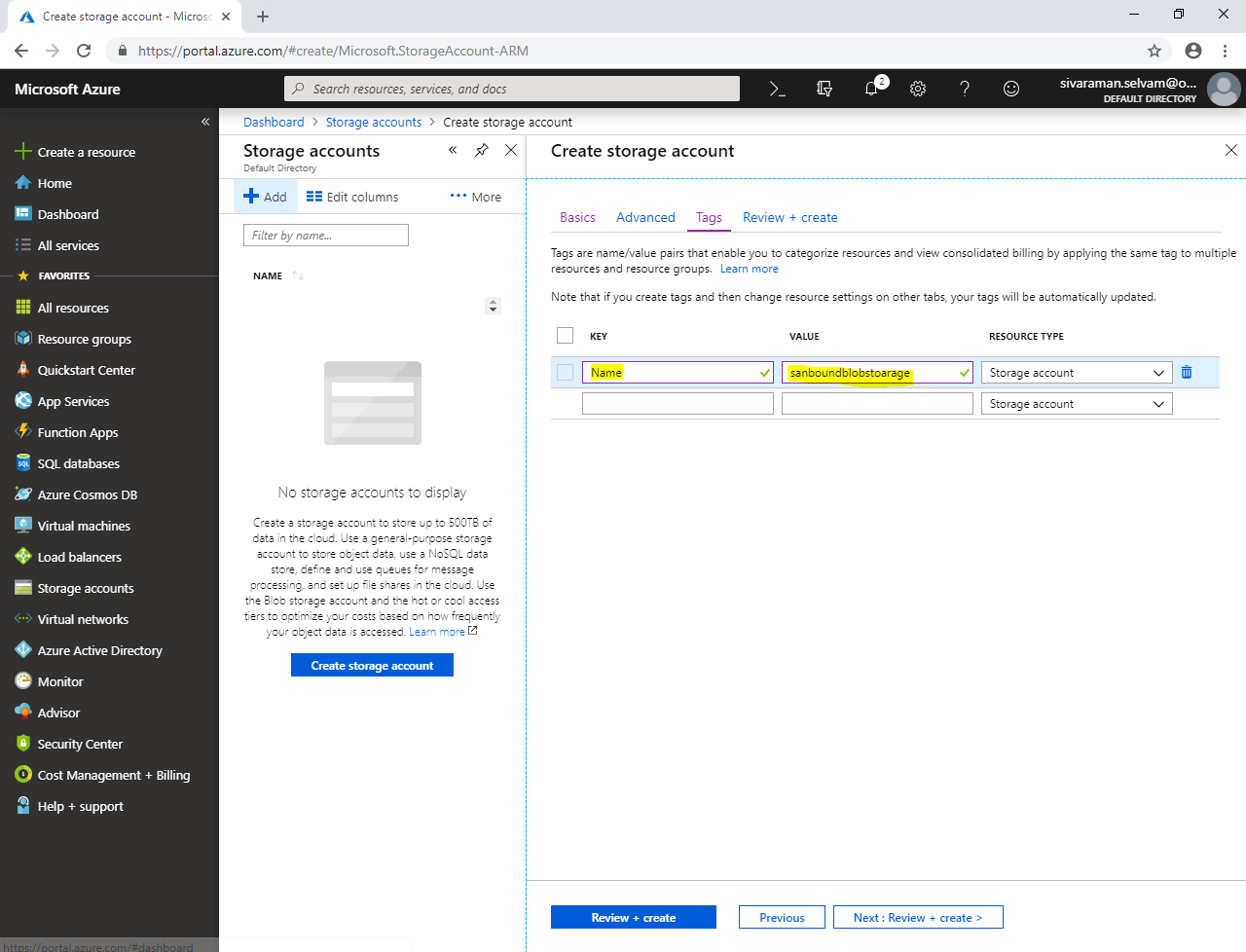
Click **“Next : Tags >”**.



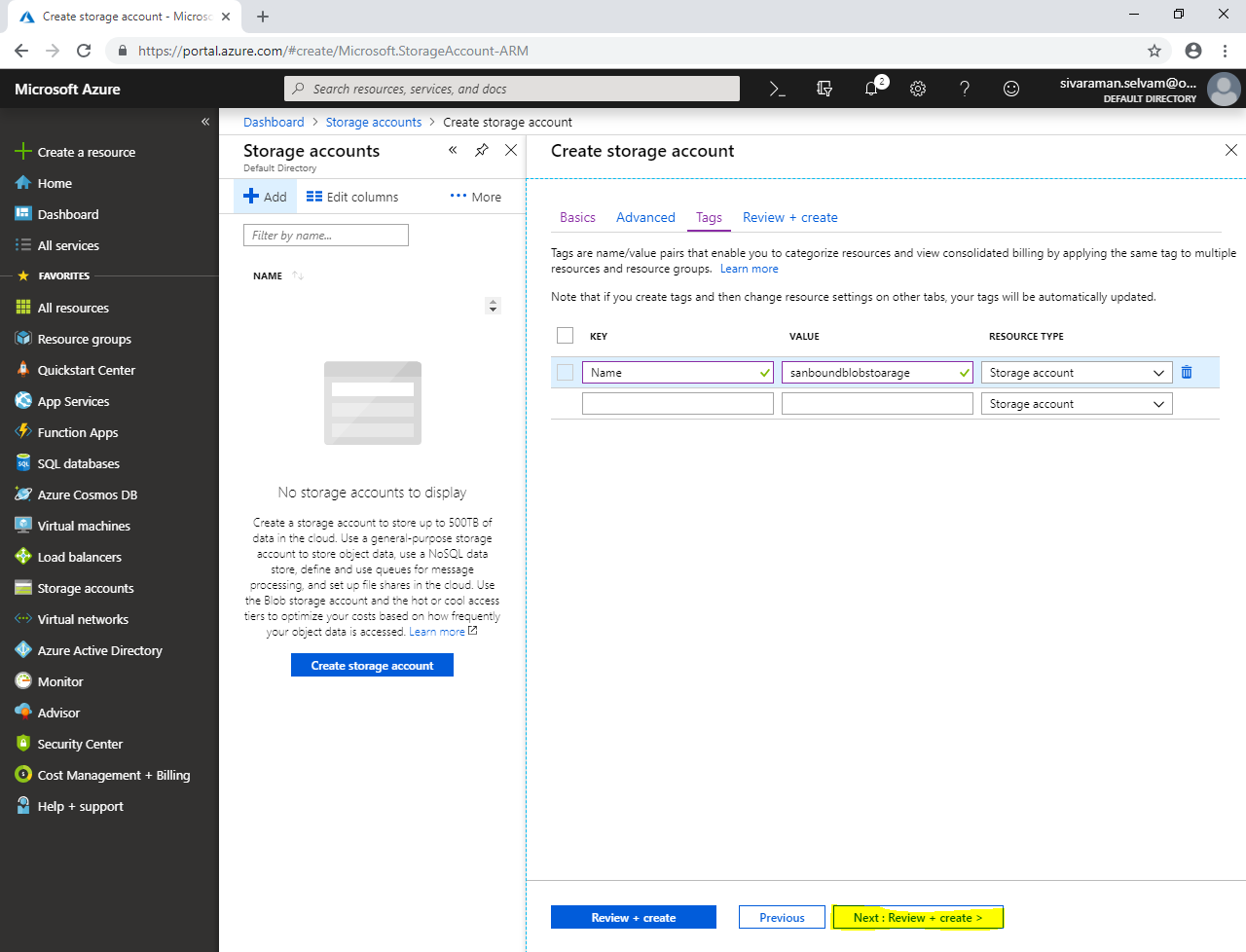
In **“Tags”**,

Type **“Key”** value as **“Name”**.

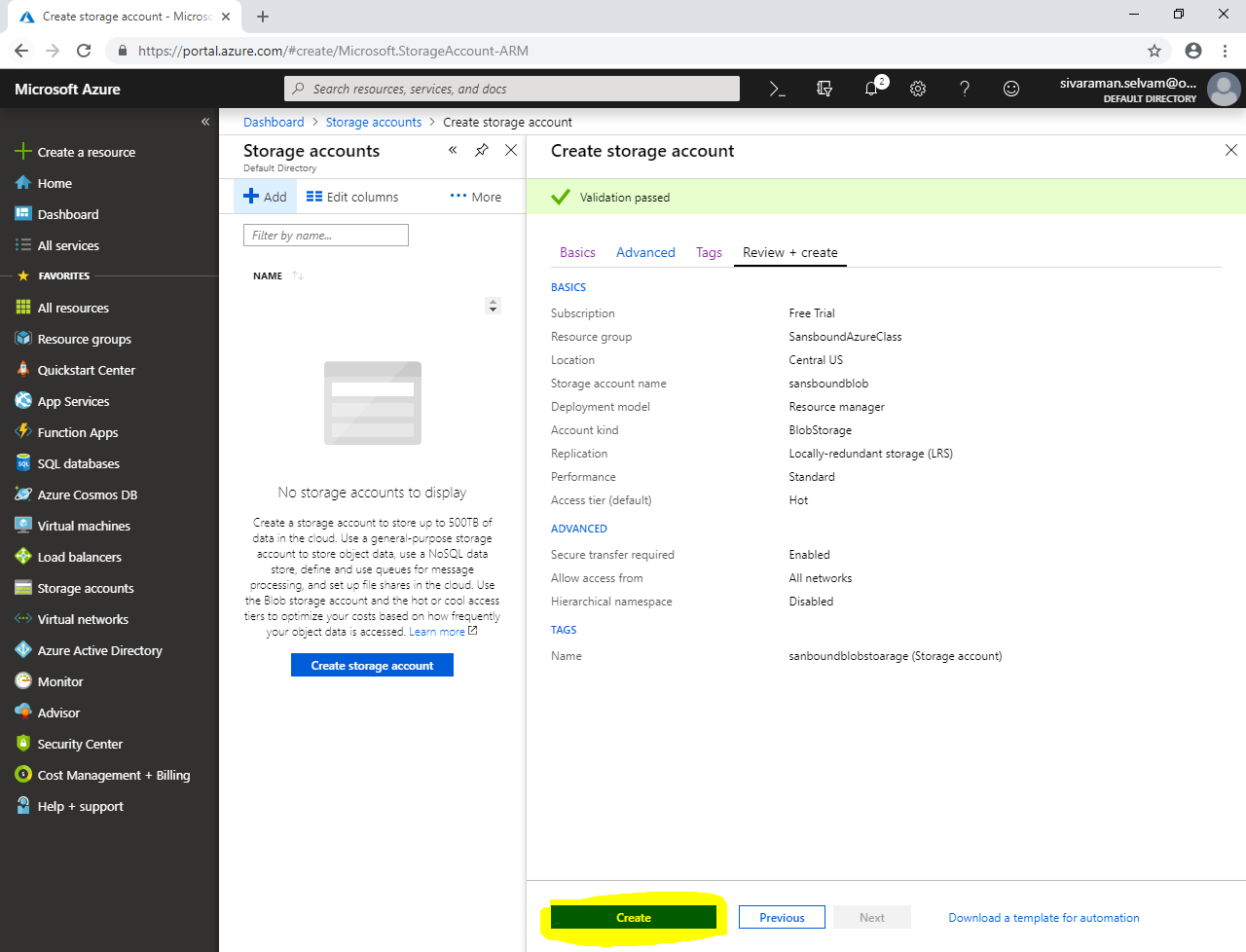
Type **“VALUE”** as **“sansboundblobstorage”**.



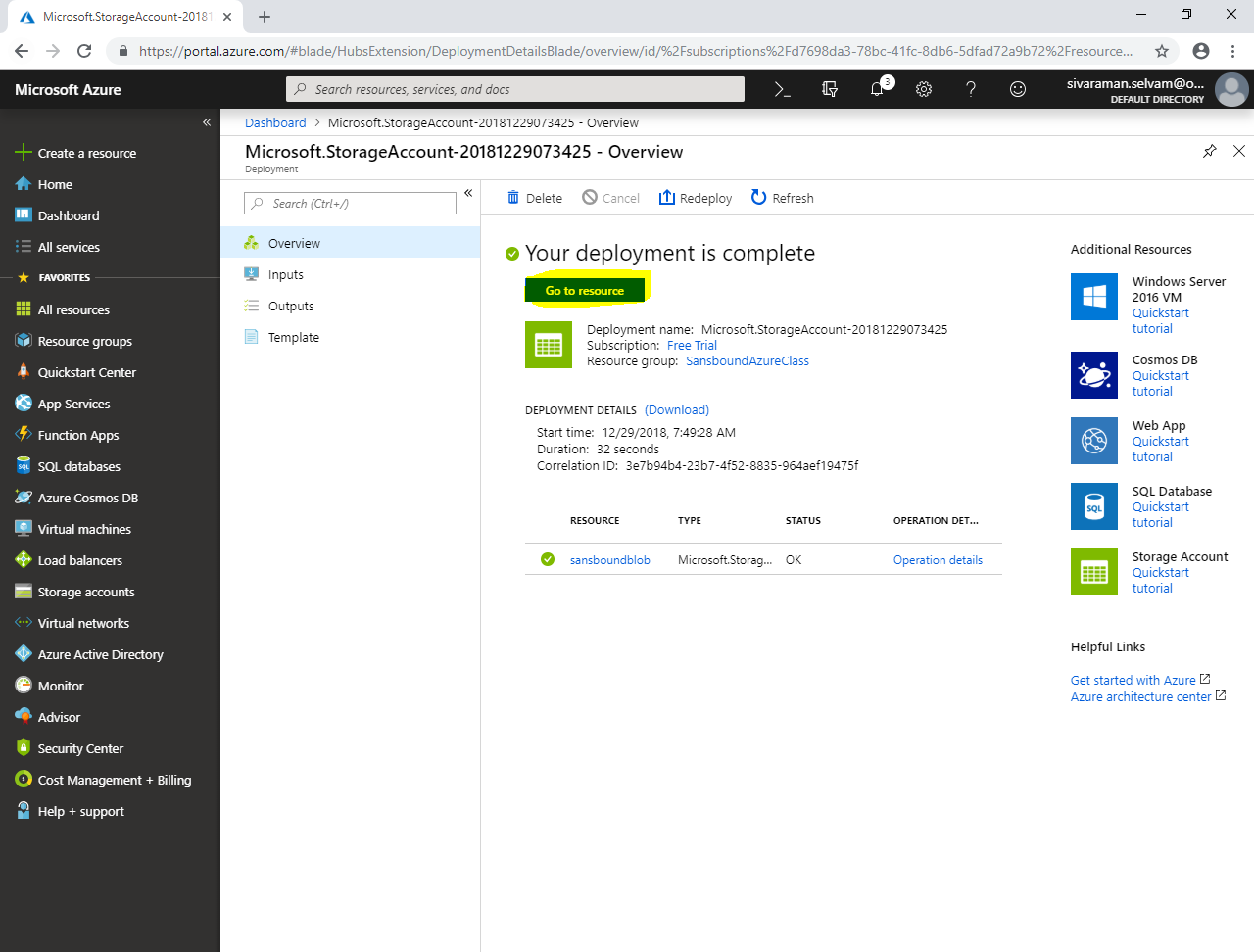
Click **“Next : Review + create >”**.



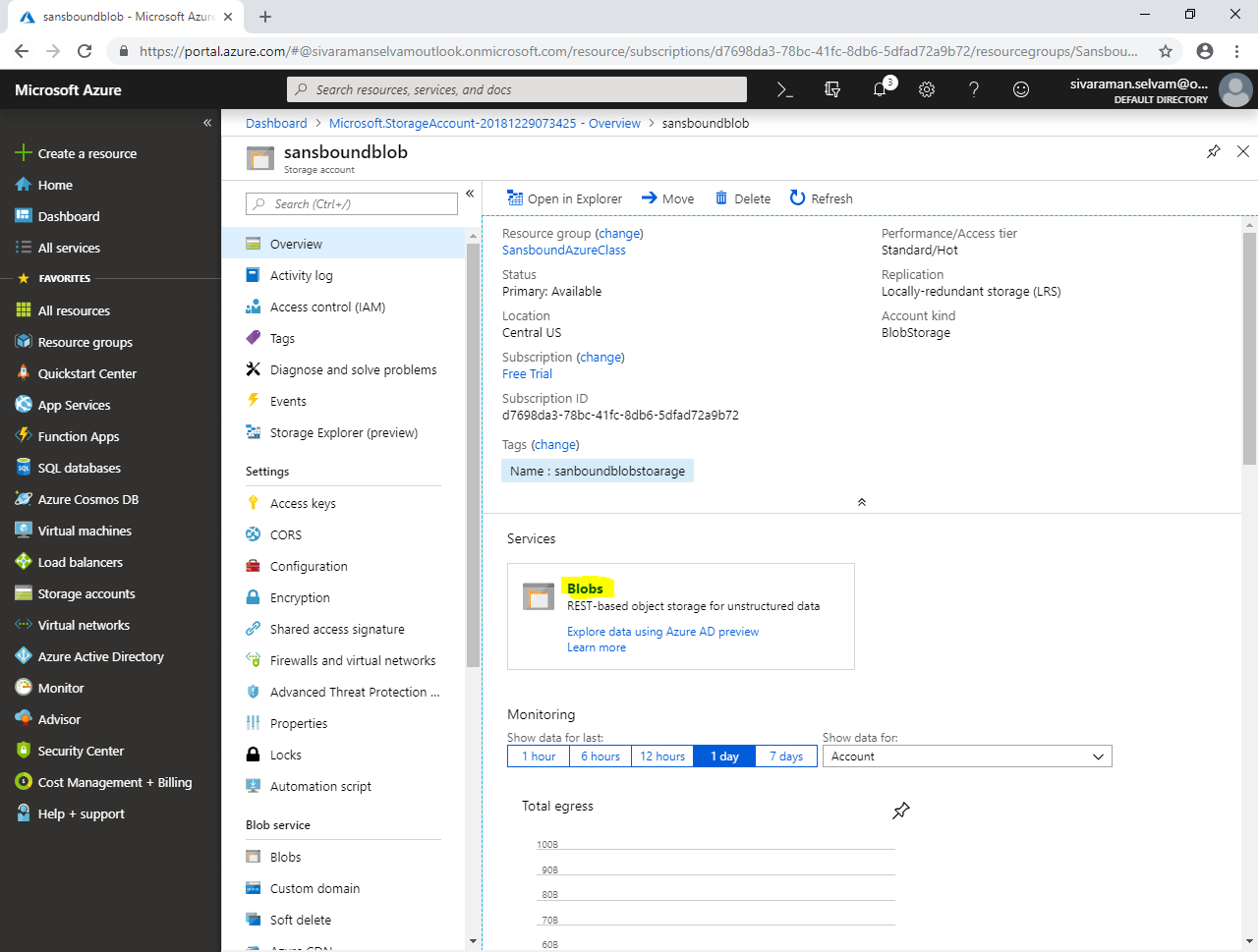
Click **“Create”**.



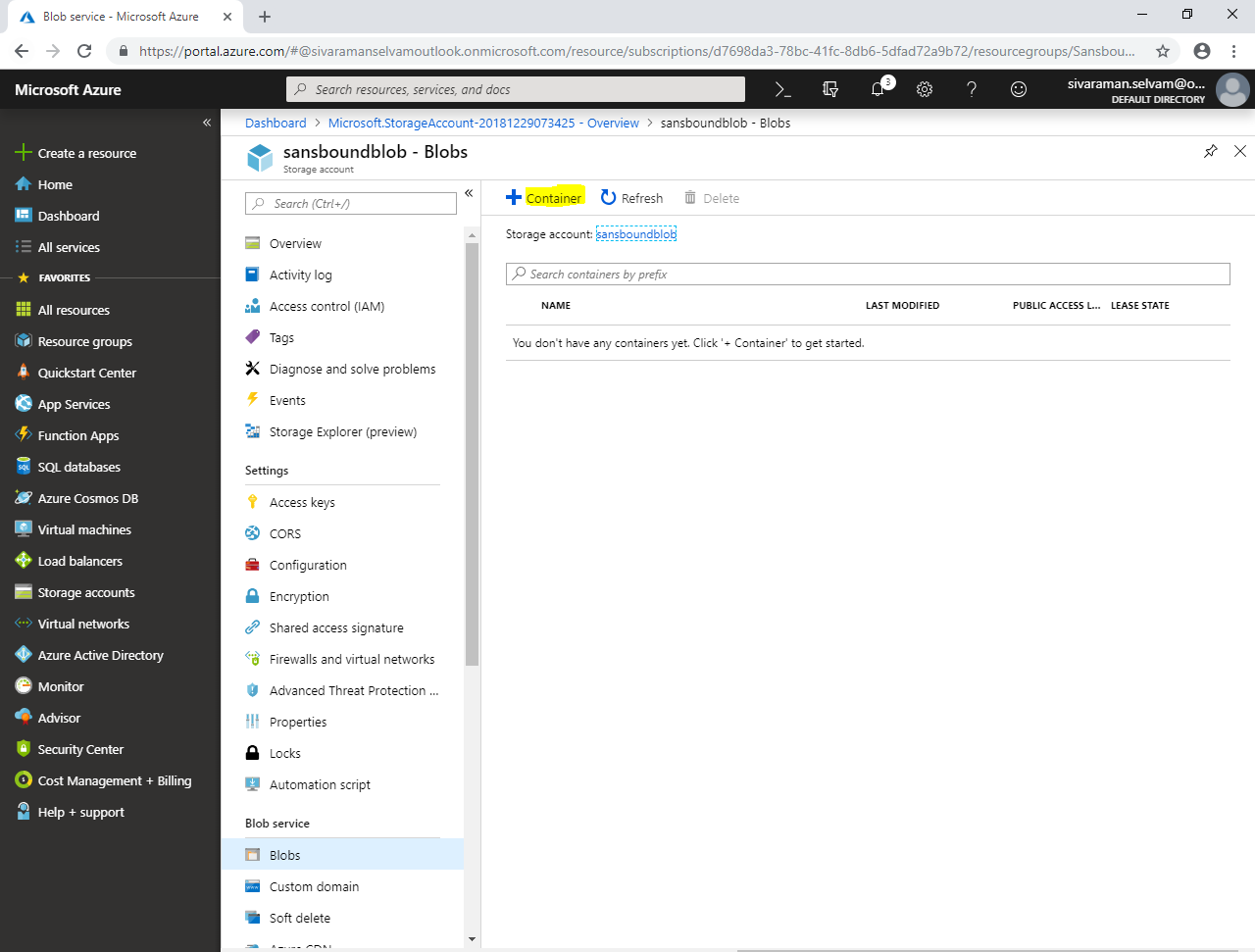
Click **“Go to resource”**.



Click **“Blobs”**.



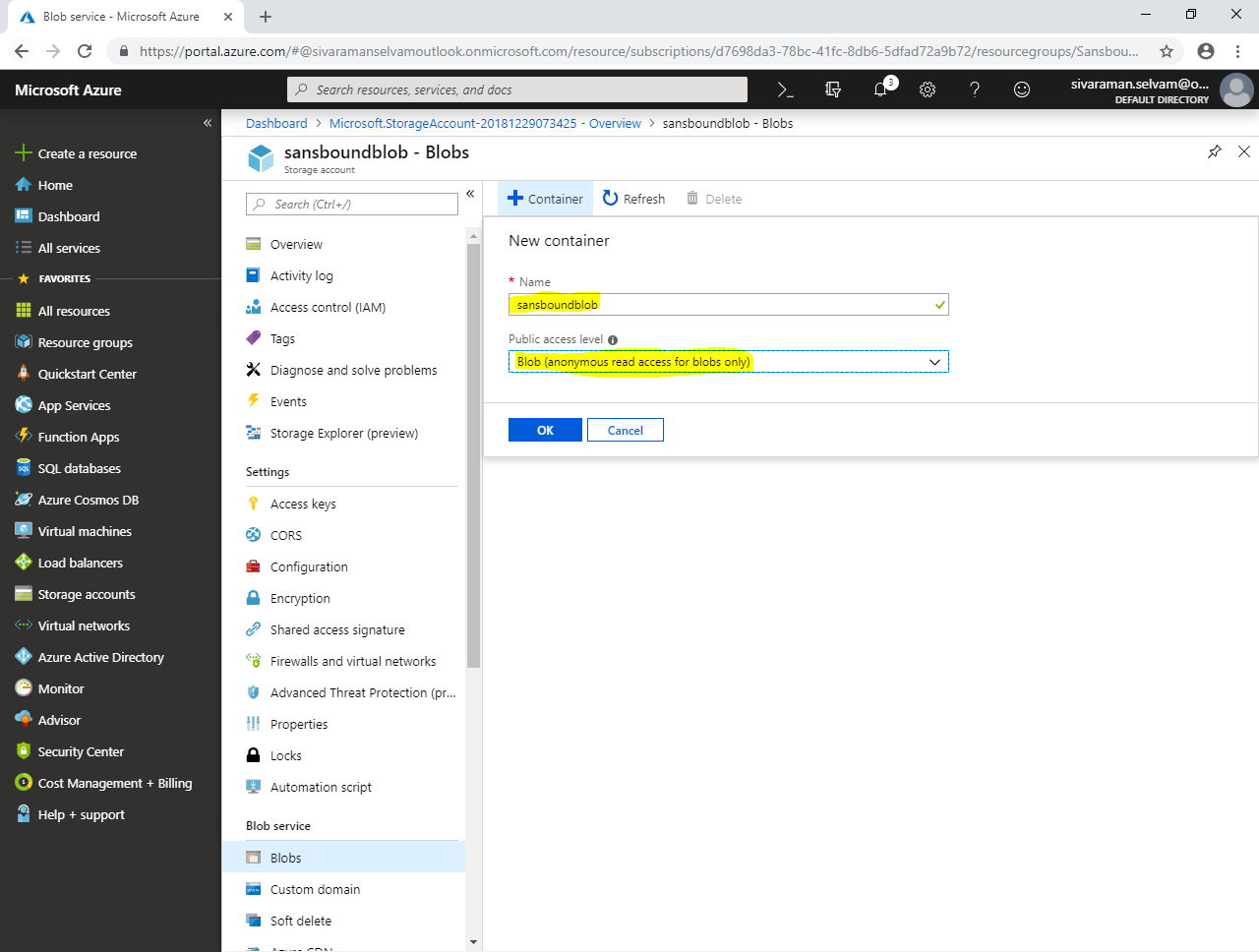
In Container, click **“Container”** to add.



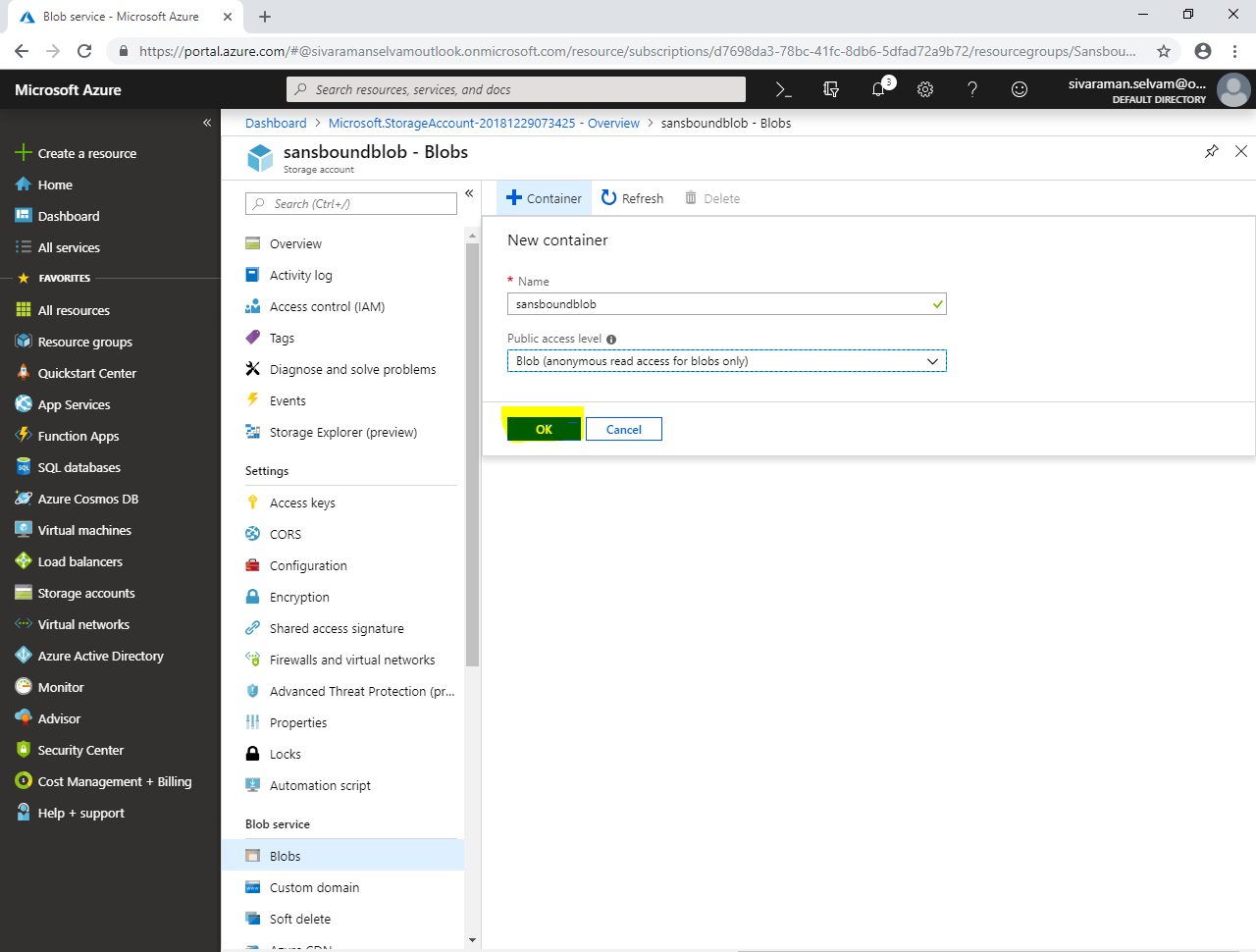
While create container,

Type **“Name”** as **“sansboundblob”**.

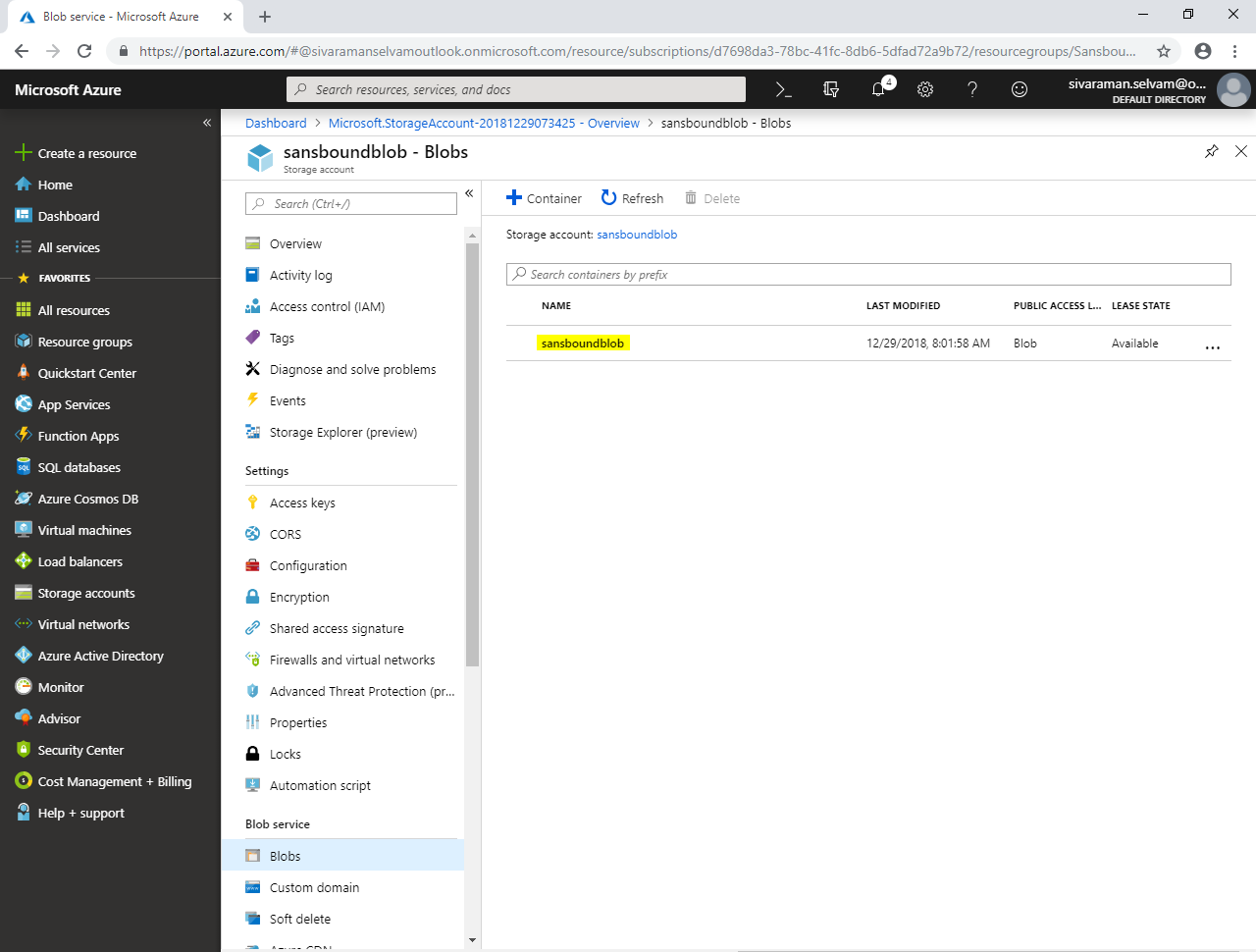
Select **“Public access level”** as **“Blob”**.



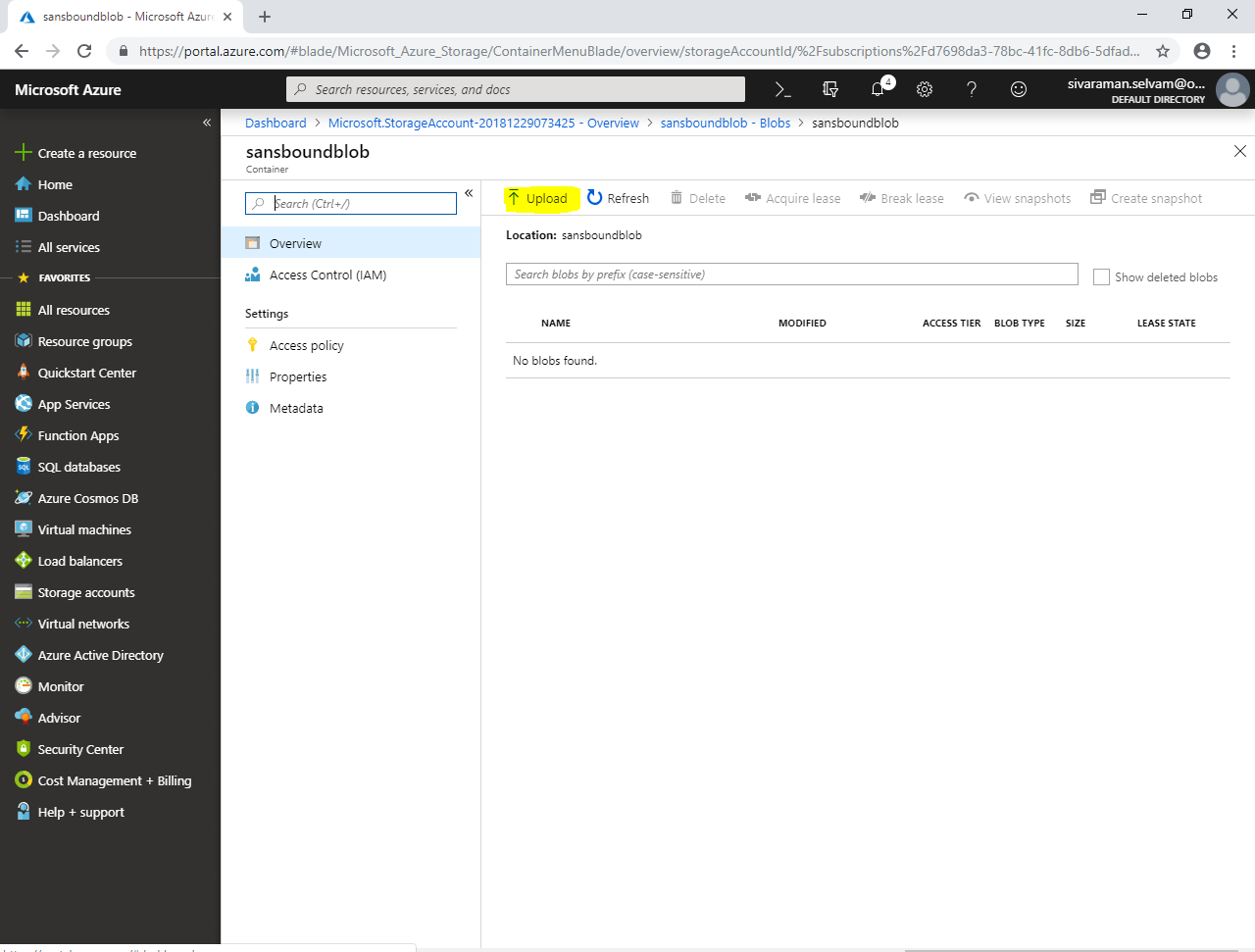
Click **“Ok”**.



Click **“sansboundblob”**.

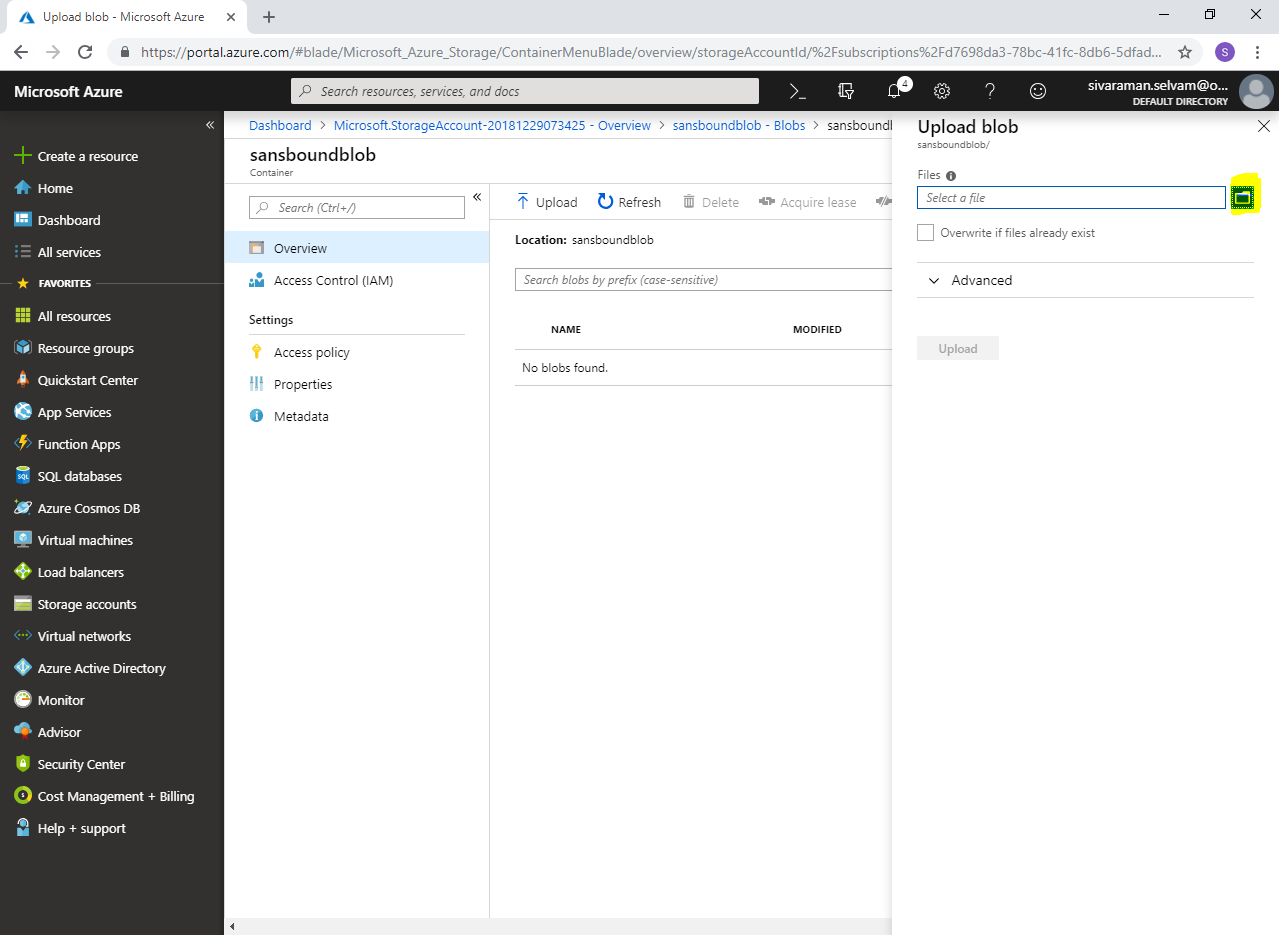


Click **“Upload”** to upload **“index.html”** file.



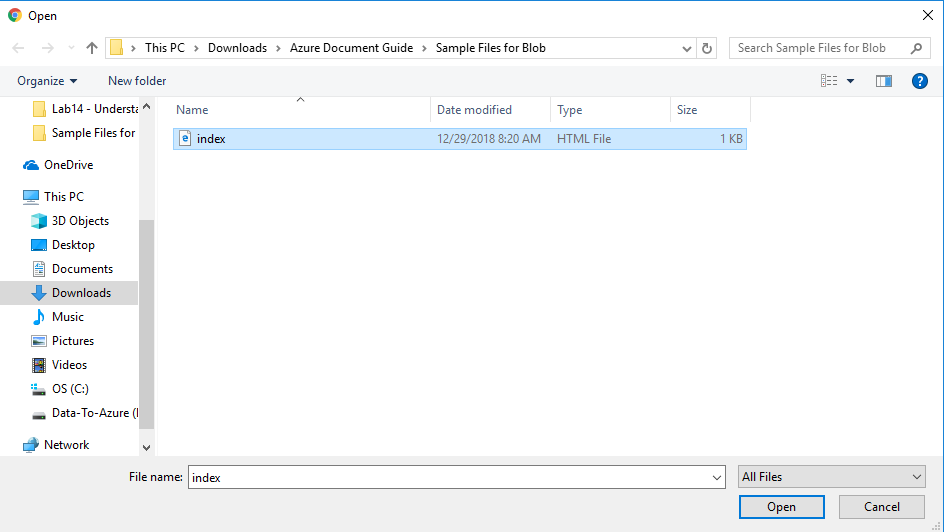
In **“Upload blob”**

Click **“Icon”**.

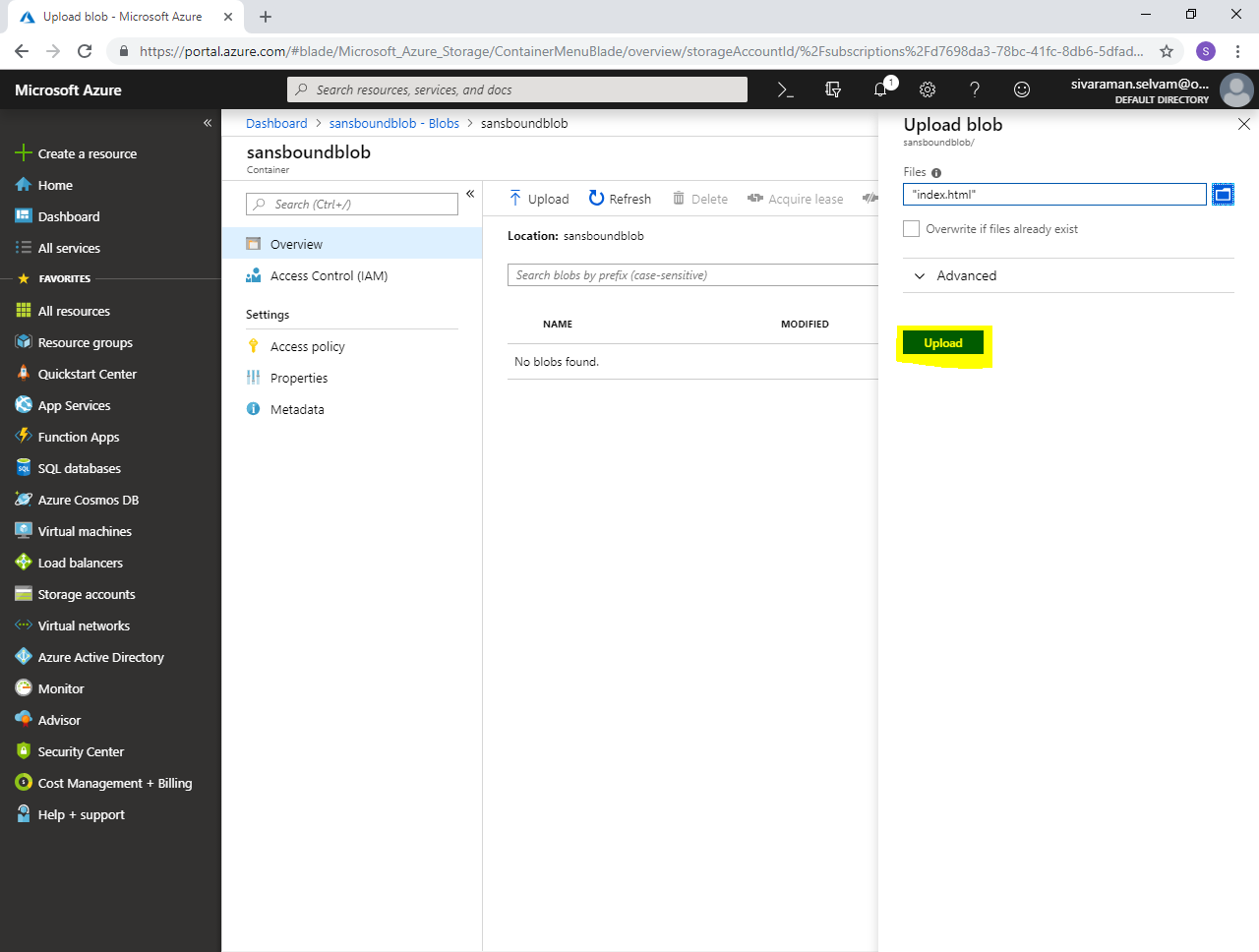


In your local machine, locate the “index.html” and select “index.html” file.

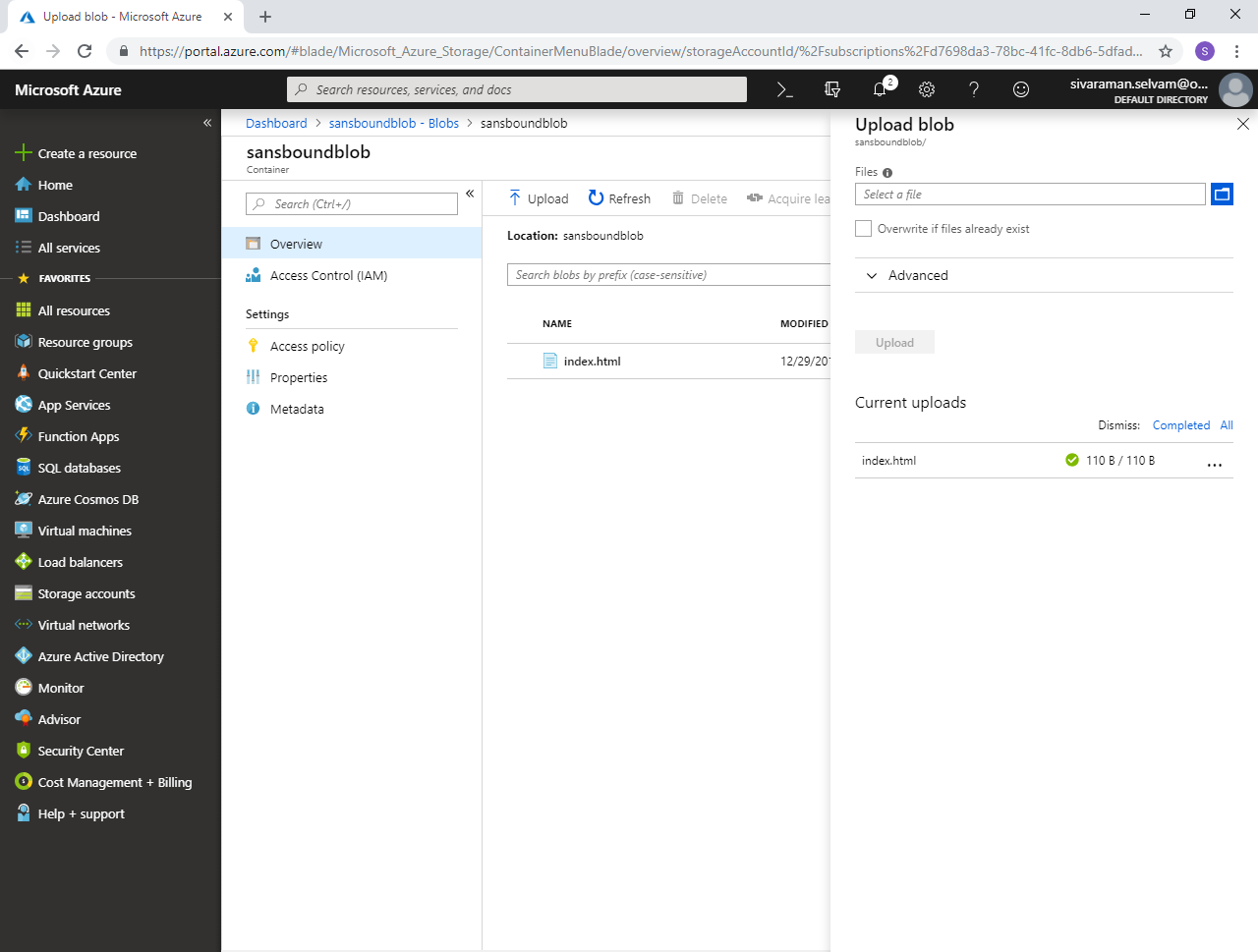
Click **“Open”**.



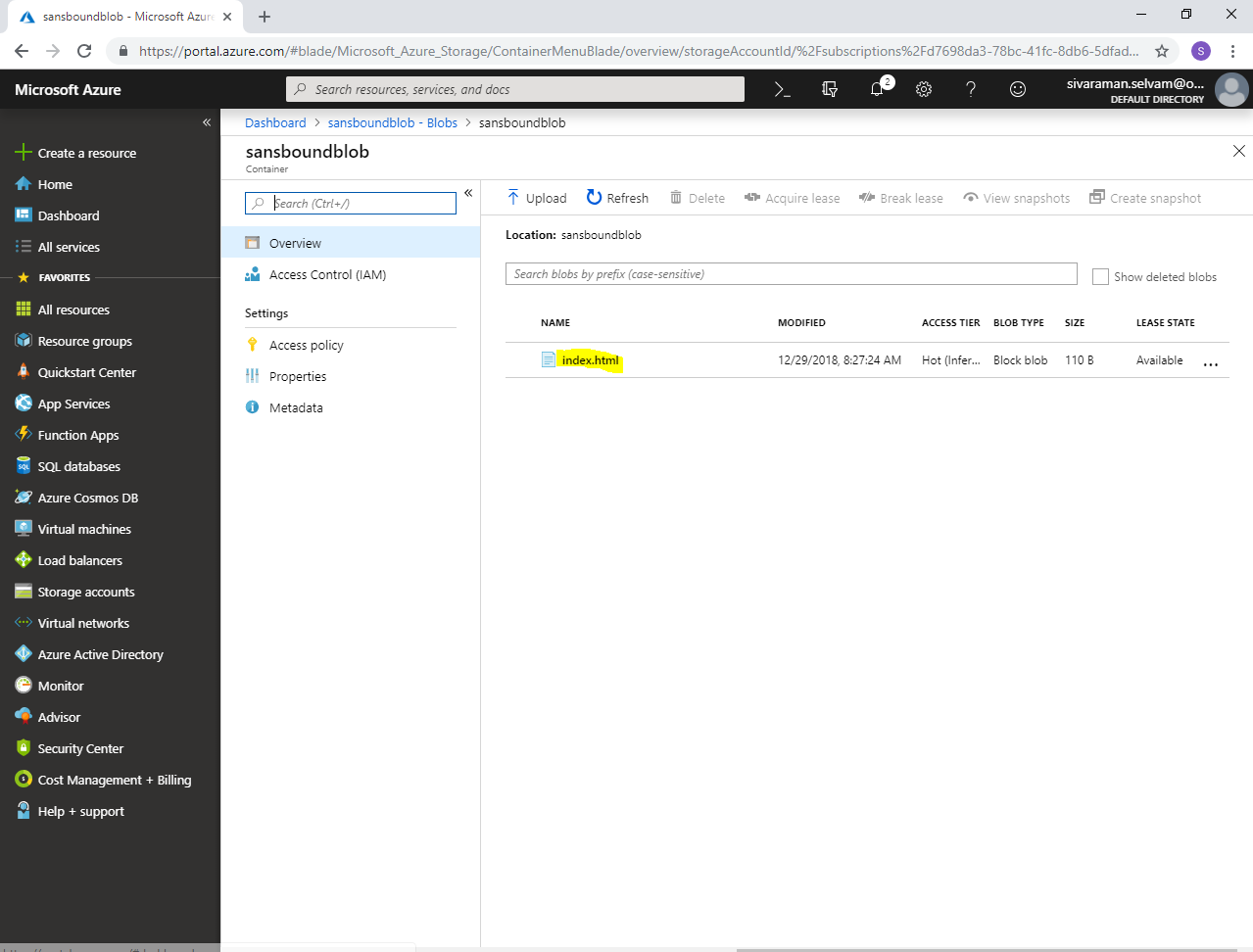
Click **“Upload”**.



You have successfully uploaded the **“index.html”** in sansboundblob container.

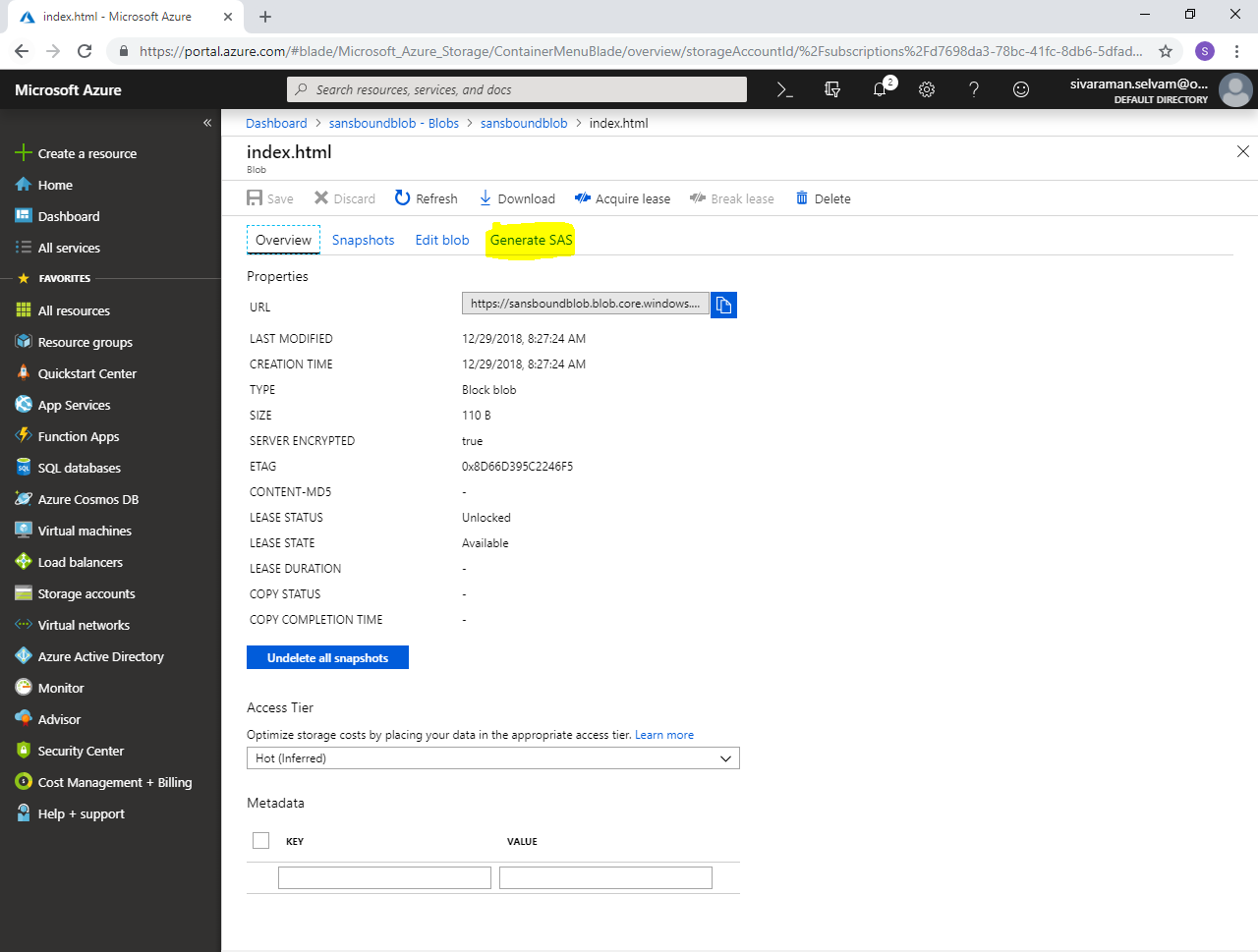


Click **“index.html”**.



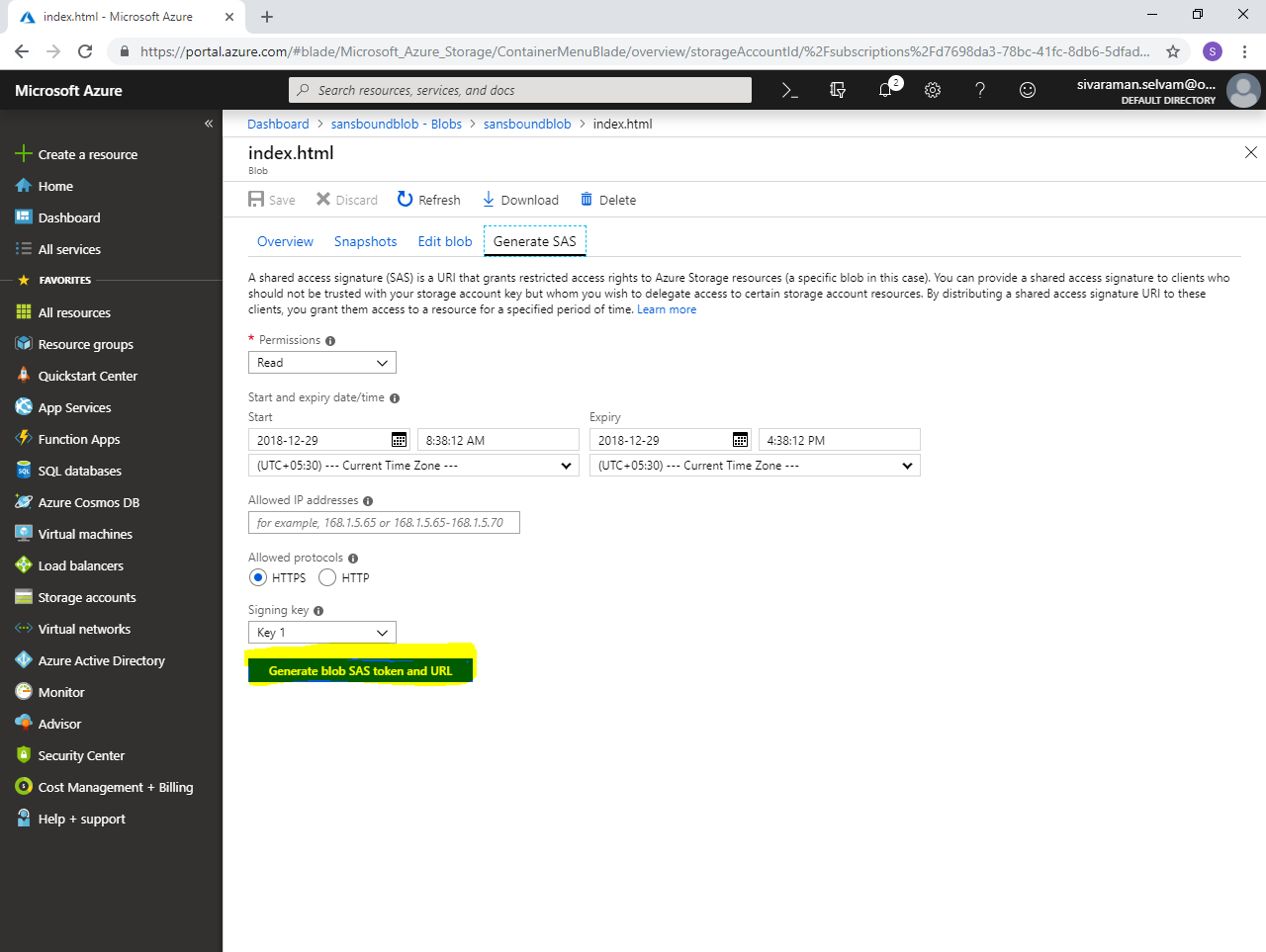
In **“Overview”**.

Click **“Generate SAS”**.



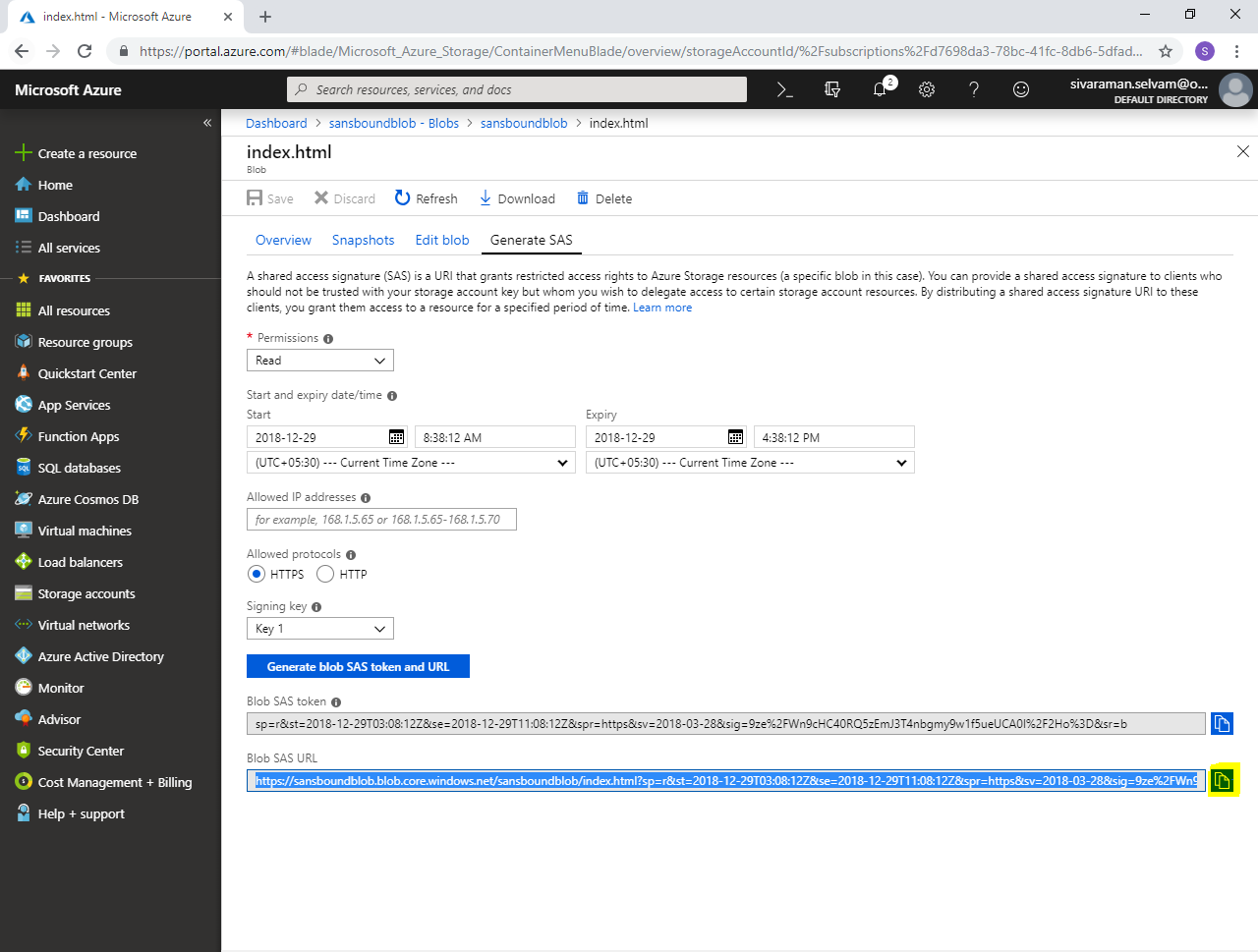
In **“Generate SAS”**.

Click **“Generate blob SAS token and URL”**.

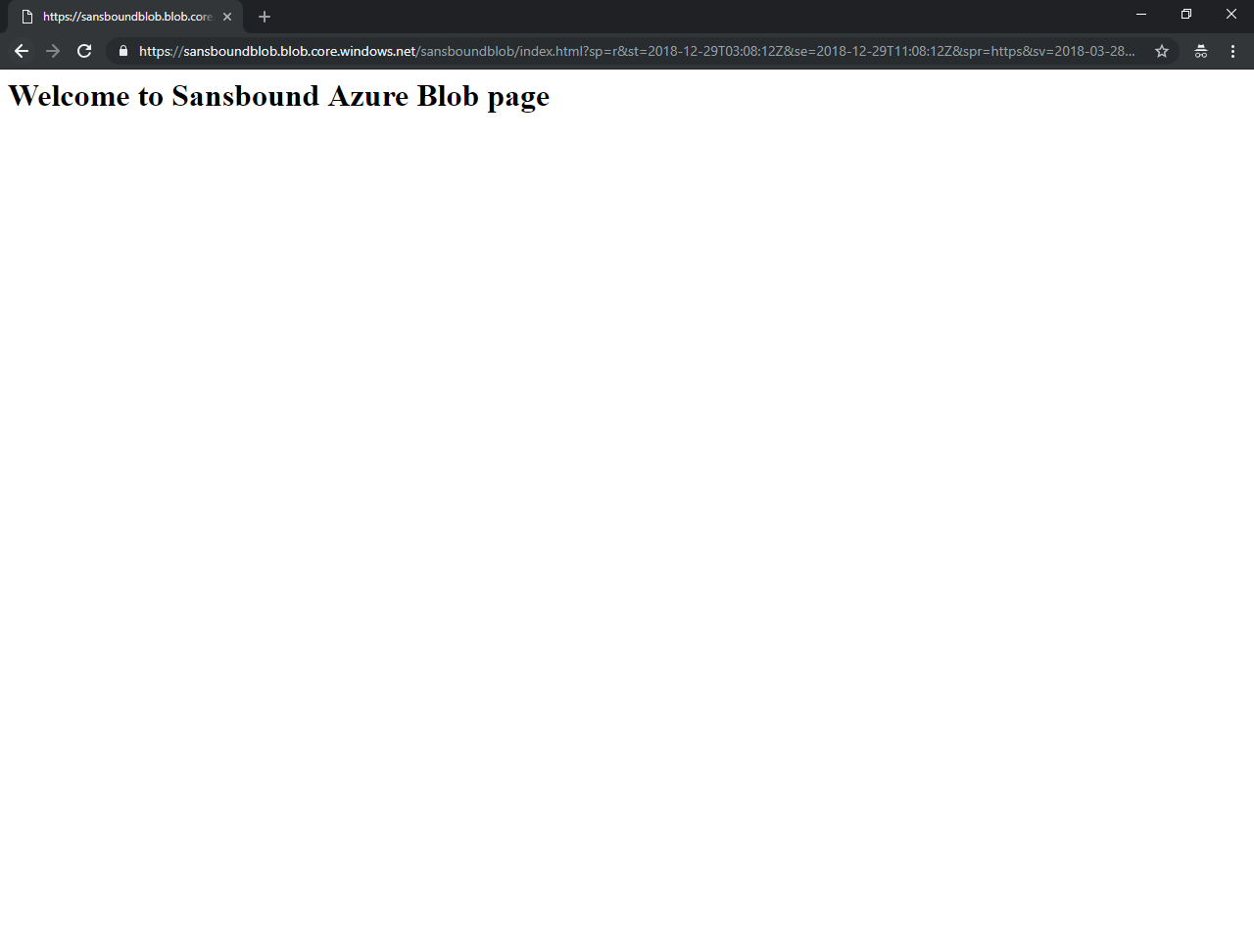


To copy **“Blob SAS URL”**.

Click **“Icon”** to copy **“Blob SAS URL”.**



Paste the “Blob SAS URL” in browser and press **“Enter”**.



You have got Blob page successfully.