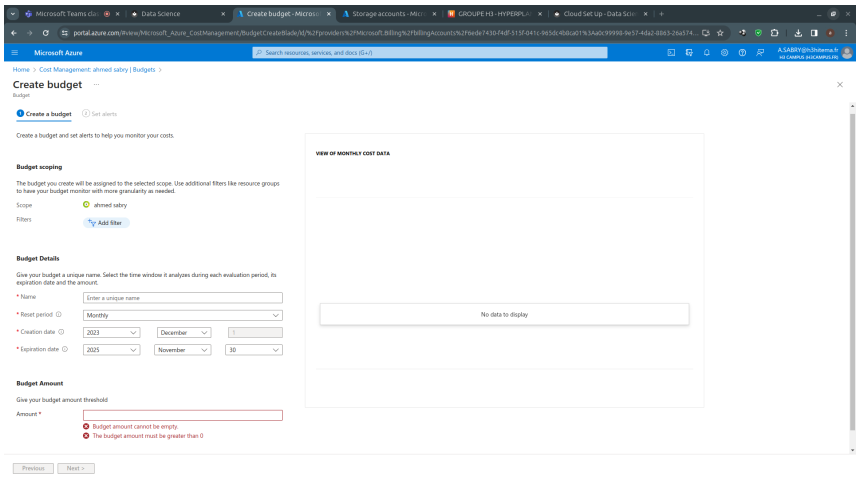
Configure Monitoring and Logging :

Access Azure Cost Management
Login to Azure: Visit the Azure home page and log in to your Azure account.

Navigate to Cost Management: Click on the "Cost Management" option in the Azure portal.

Setting Up Budgets
Access Budgets: Within Cost Management, go to the "Budgets" section.



Home > Cost Management: ahmed sabry | Budgets >

Create budget
Budget

1 Create a budget   2 Set alerts

Create a budget and set alerts to help you monitor your costs.

Budget scoping

The budget you create will be assigned to the selected scope. Use additional filters like resource groups to have your budget monitor with more granularity as needed.

Scope	ahmed sabry
Filters	Add filter

Budget Details

Give your budget a unique name. Select the time window it analyzes during each evaluation period, its expiration date and the amount.

* Name	Enter a unique name		
* Reset period	Monthly		
* Creation date	2023	December	1
* Expiration date	2025	November	30

Budget Amount

Give your budget amount threshold

Amount *

Budget amount cannot be empty.
The budget amount must be greater than 0

VIEW OF MONTHLY COST DATA

No data to display

Create a New Budget: Select "Add" to start configuring a new budget for monitoring costs.

Define Budget Details
Name Your Budget: Provide a unique and descriptive name for the budget.

Choose Time Period: Select the budget duration—monthly, quarterly, or annually.

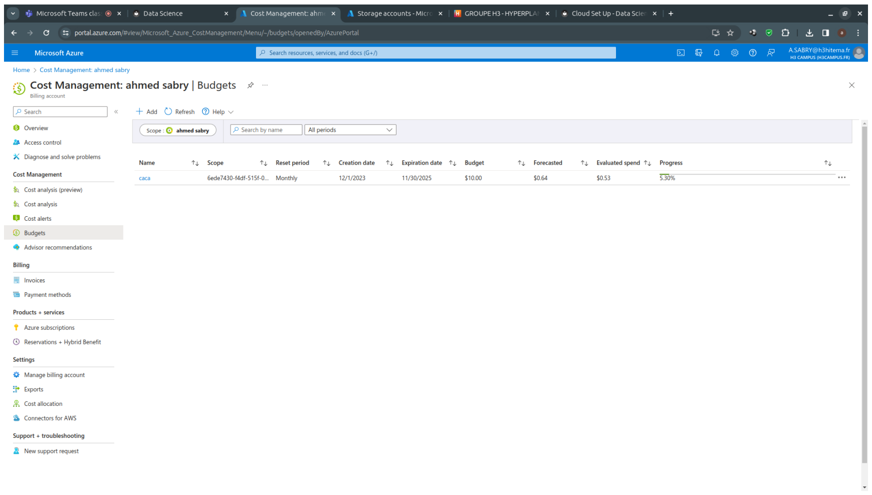
Alert Type: Decide whether to receive alerts based on actual expenses or forecasted costs.

Set Alert Thresholds
Specify Budget Limits: Define the threshold amount that, when exceeded, triggers an alert.

Enter Alert Recipient: Add the email address where notifications will be sent when the threshold is crossed.

Language Preference: Choose the language for alert notifications.

Activate the Budget
Review & Confirm: Double-check all details, ensuring accuracy.
Create the Budget: Click on "Create" to finalize and activate your budget settings.



portal.azure.com/#view/Microsoft_Azure_CostManagement/Menu/~/budgets/openedBy/AzurePortal

Cost Management: ahmed sabry | Budgets
Billing account

Scope : ahmed sabry

Name	Scope	Reset period	Creation date	Expiration date	Budget	Forecasted	Evaluated spend	Progress
caca	6ede7430-f4df-515f-0...	Monthly	12/1/2023	11/30/2025	$10.00	$0.64	$0.53	5.30%

Implement Azure Identity by creating a simple dev role :

Navigate to Resource Groups

Search for Resource Groups: In the Azure Portal's search bar, type "Resource groups" and select it from the suggestions.

Create a New Resource Group

Click on 'Add': Within the Resource Groups section, click on the "Add" button to initiate creating a new resource group.

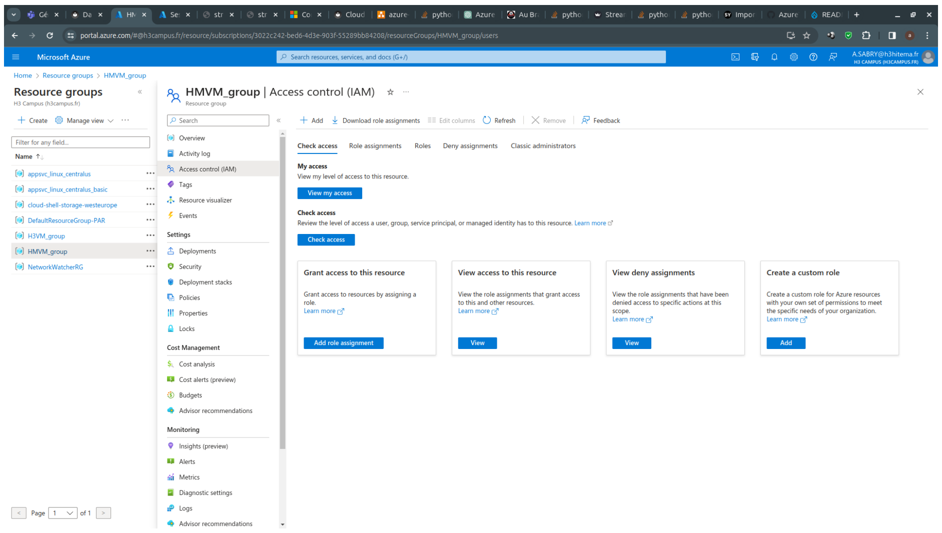
Provide Resource Group Details

Subscription: Choose the Azure subscription to which this resource group will belong (if you have multiple subscriptions).
Resource Group Details:

- Name: Enter a unique name for your resource group, following Azure naming conventions (e.g., "MyProjectResourceGroup").
- Region: Choose the Azure region where you want the resource group to be located.

Review & Create:

- Review Configuration: Double-check the details you've provided.
- Create the Resource Group: Click on the "Review + Create" button.



My access
View my level of access to this resource.

Check access
Review the level of access a user, group, service principal, or managed identity has to this resource. Learn more

Grant access to this resource
Grant access to resources by assigning a role.
Learn more

View access to this resource
View the role assignments that grant access to this and other resources.
Learn more

View deny assignments
View the role assignments that have been denied access to specific actions at this scope.
Learn more

Create a custom role
Create a custom role for Azure resources with your own set of permissions to meet the specific needs of your organization.
Learn more

Confirmation & Access

Validation: Azure will perform a validation check to ensure all details are correct.
Create the Resource Group: Once validated, click on the "Create" button to finalize the creation of your personalized resource group.

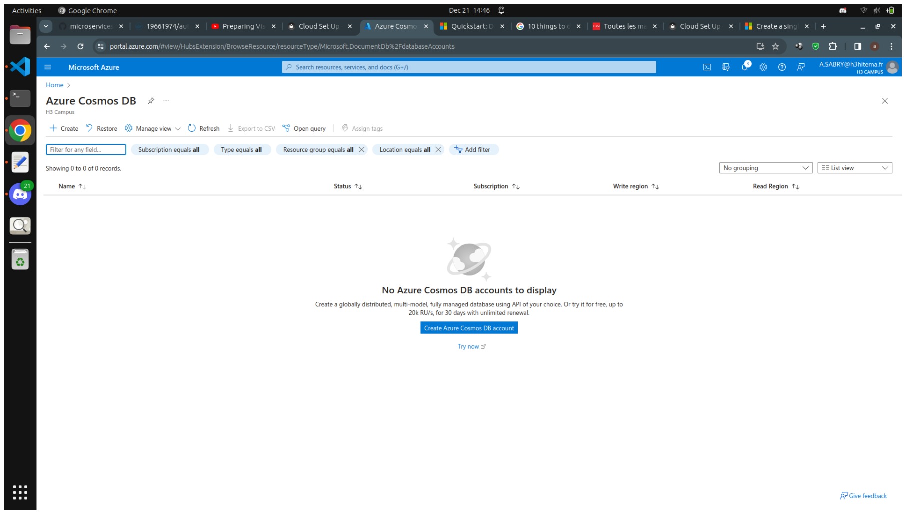
Accessing Your Resource Group

Access Resource Group: Navigate back to the Resource Groups section and search for or select your newly created resource group.

Managing Resources within the Resource Group

Add Resources: To add resources, click on "+ Add" within your resource group and select the type of resource you want to add (e.g., Virtual Machine, Storage Account, etc.)

Steps to Create a Cosmos DB MongoDB Account in Azure Portal:



portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.DocumentDb%2FdatabaseAccounts

Azure Cosmos DB
H3 Campus

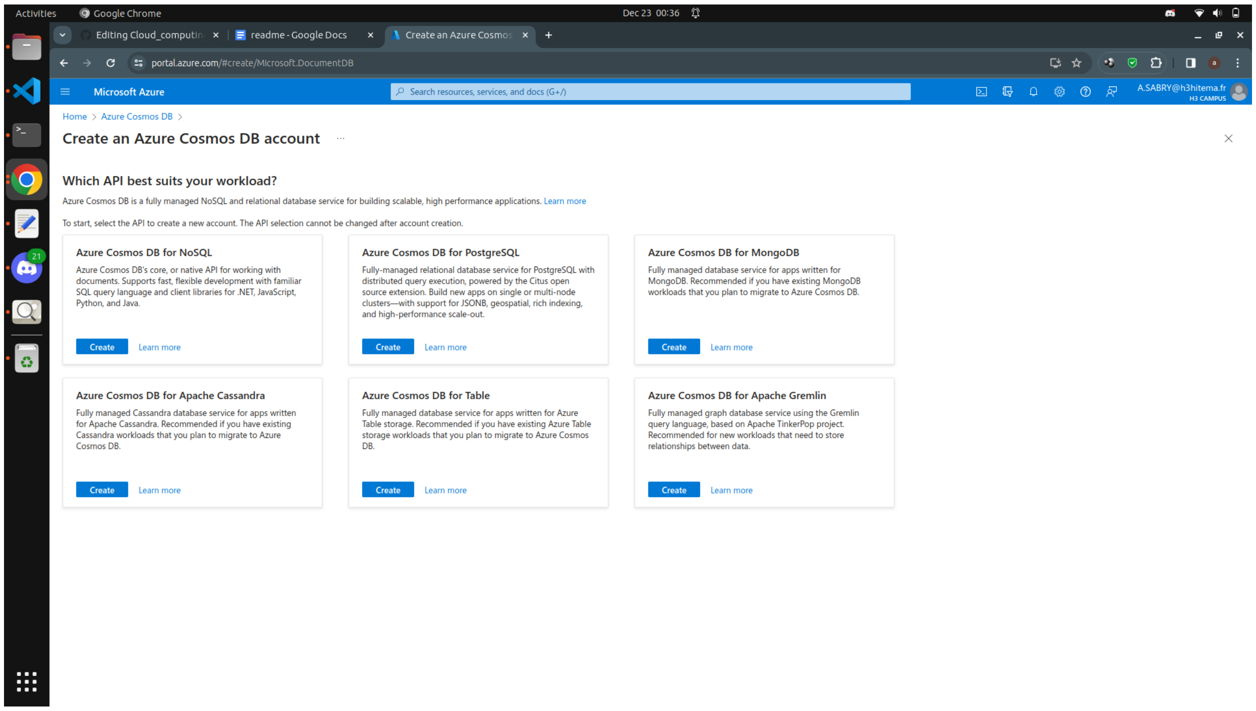
Showing 0 to 0 of 0 records.

Name	Status	Subscription	Write region	Read Region

No Azure Cosmos DB accounts to display

Create a globally distributed, multi-model, fully managed database using API of your choice. Or try it for free, up to 20k RU/s, for 30 days with unlimited renewal.

- Click on the "+ Create a resource" button on the upper left-hand corner.
- In the search bar, type "Azure Cosmos DB" and select it from the results.
- Click "Create" to start configuring the Cosmos DB account.



portal.azure.com/#create/Microsoft.DocumentDB

Home > Azure Cosmos DB >

Create an Azure Cosmos DB account

Which API best suits your workload?

Azure Cosmos DB is a fully managed NoSQL and relational database service for building scalable, high performance applications. Learn more

To start, select the API to create a new account. The API selection cannot be changed after account creation.

Azure Cosmos DB for NoSQL
Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

Azure Cosmos DB for PostgreSQL
Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

Azure Cosmos DB for MongoDB
Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for Apache Cassandra
Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for Table
Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for Apache Gremlin
Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.

Basic Settings:

- Choose the Subscription and Resource Group where you want to create the Cosmos DB account.
- Provide an Account name (this needs to be a unique name) and select the API as "Azure Cosmos DB for MongoDB" as the API.
- Choose the desired Location (where your data will be stored).

Additional Configuration:

- Under the "Account Details" section, choose the API version, capacity mode, and any additional options you require.
- Select the appropriate options for Availability, Consistency, and Multi-region Writes based on your needs.

Networking:

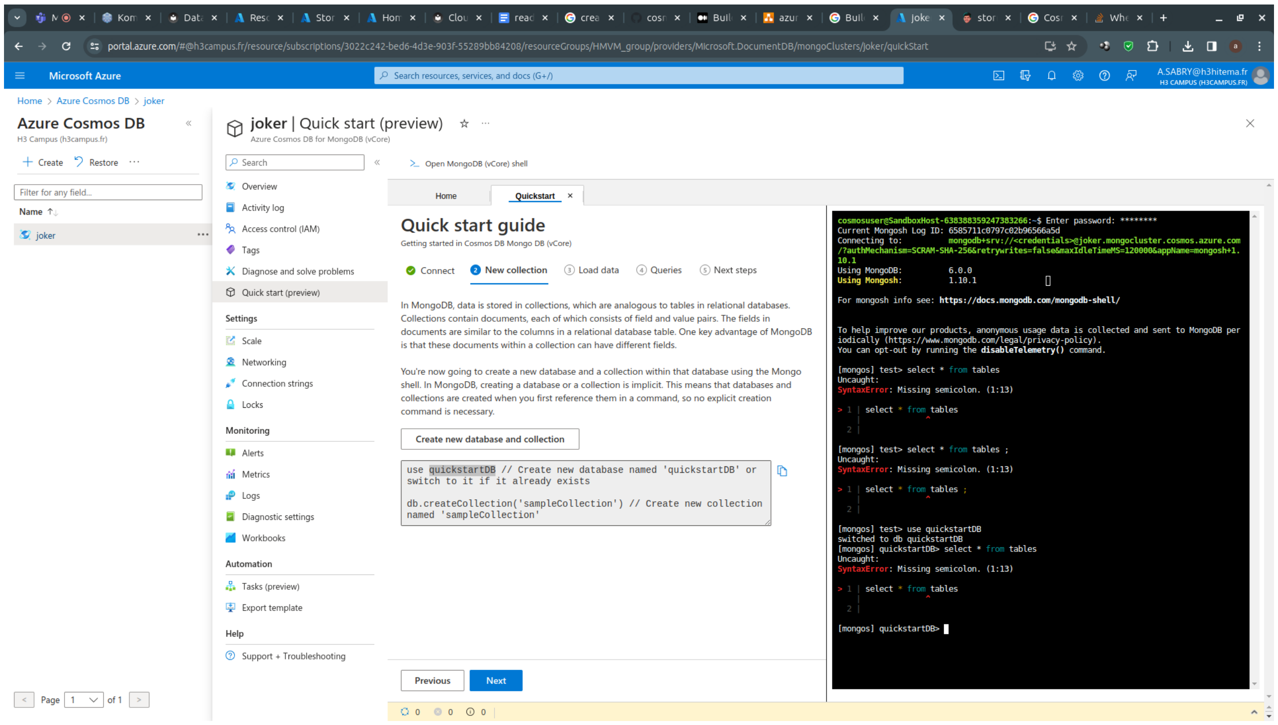
- Configure the network settings according to your security and accessibility preferences (firewalls, virtual networks, etc.).

Review + Create:

- Review your configurations to ensure they're accurate.
- Click "Create" to start deploying the Cosmos DB MongoDB account.

Access and Manage:

- Once deployment is complete, navigate to your newly created Cosmos DB account in the Azure Portal.
- Access the account's settings, where you'll find connection strings and other configuration details required to connect to your MongoDB instance within Cosmos DB.



Quick start guide

Getting started in Cosmos DB Mongo DB (vCore)

In MongoDB, data is stored in collections, which are analogous to tables in relational databases. Collections contain documents, each of which consists of field and value pairs. The fields in documents are similar to the columns in a relational database table. One key advantage of MongoDB is that these documents within a collection can have different fields.

You're now going to create a new database and a collection within that database using the Mongo shell. In MongoDB, creating a database or a collection is implicit. This means that databases and collections are created when you first reference them in a command, so no explicit creation command is necessary.

```
use quickstartDB // Create new database named 'quickstartDB' or switch to it if it already exists

db.createCollection('sampleCollection') // Create new collection named 'sampleCollection'
```

```
cosmosuser@SandboxHost-638388359247383266:~$ Enter password: ********
Current Mongosh Log ID: 6585711c0797c02b96566a5d
Connecting to:          mongodb+srv://<credentials>@joker.mongocluster.cosmos.azure.com
/?authMechanism=SCRAM-SHA-256&retrywrites=false&maxIdleTimeMS=120000&appName=mongosh+1.
10.1
Using MongoDB:          6.0.0
Using Mongosh:          1.10.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB per
iodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

[mongos] test> select * from tables
Uncaught:
SyntaxError: Missing semicolon. (1:13)

> 1 | select * from tables
    |              ^
  2 |

[mongos] test> select * from tables ;
Uncaught:
SyntaxError: Missing semicolon. (1:13)

> 1 | select * from tables ;
    |              ^
  2 |

[mongos] test> use quickstartDB
switched to db quickstartDB
[mongos] quickstartDB> select * from tables
Uncaught:
SyntaxError: Missing semicolon. (1:13)

> 1 | select * from tables
    |              ^
  2 |

[mongos] quickstartDB>
```

Steps to Create a Storage Account in Azure:

Sign in to Azure Portal:

- Go to portal.azure.com and log in to your Azure account.

Create a Storage Account:

- Click on "+ Create a resource" in the upper left-hand corner.
- Search for "Storage account" and select it from the results.
- Click "Create" to start configuring your Storage Account.

Configure Storage Account Settings:

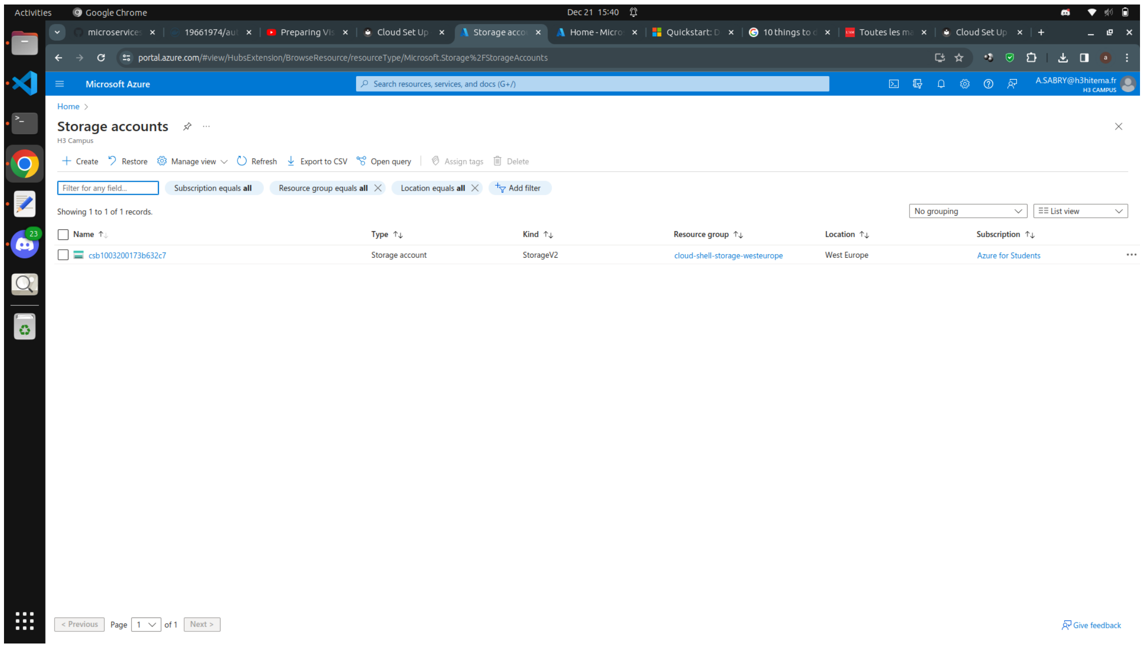
- Choose your Subscription and Resource Group.
- Provide a unique name for your Storage Account (naming rules apply).
- Select the desired Location for your storage data.
- Choose the Performance tier (Standard or Premium).
- Define the Account kind (StorageV2, BlobStorage, FileStorage, etc.).
- Set the Replication option (Locally redundant storage (LRS), Zone-redundant storage (ZRS), Geo-redundant storage (GRS), etc.).
- Configure other settings like Access tiers, Network settings, etc., based on your requirements.

Review + Create:

- Review your configurations to ensure accuracy.
- Click "Create" to deploy your Storage Account.

Access and Manage Storage Account:

- Once the deployment is complete, navigate to your newly created Storage Account in the Azure Portal.
- Access the various components and features of your Storage Account such as Containers, File Shares, Queues, Tables, etc.



Storage accounts

H3 Campus

Showing 1 to 1 of 1 records.

Name	Type	Kind	Resource group	Location	Subscription
csb1003200173b632c7	Storage account	StorageV2	cloud-shell-storage-westeurope	West Europe	Azure for Students

Additional Actions with Storage Accounts:

Access Keys and Connection Strings:

- Obtain Access Keys or Shared Access Signatures (SAS) to access your storage account programmatically.
- Use these keys or connection strings in your applications to interact with the Azure Storage services.

Configure Security and Access Controls:

- Set up access controls using Azure RBAC (Role-Based Access Control) to manage who has access to your Storage Account and what they can do.

Monitoring and Management:

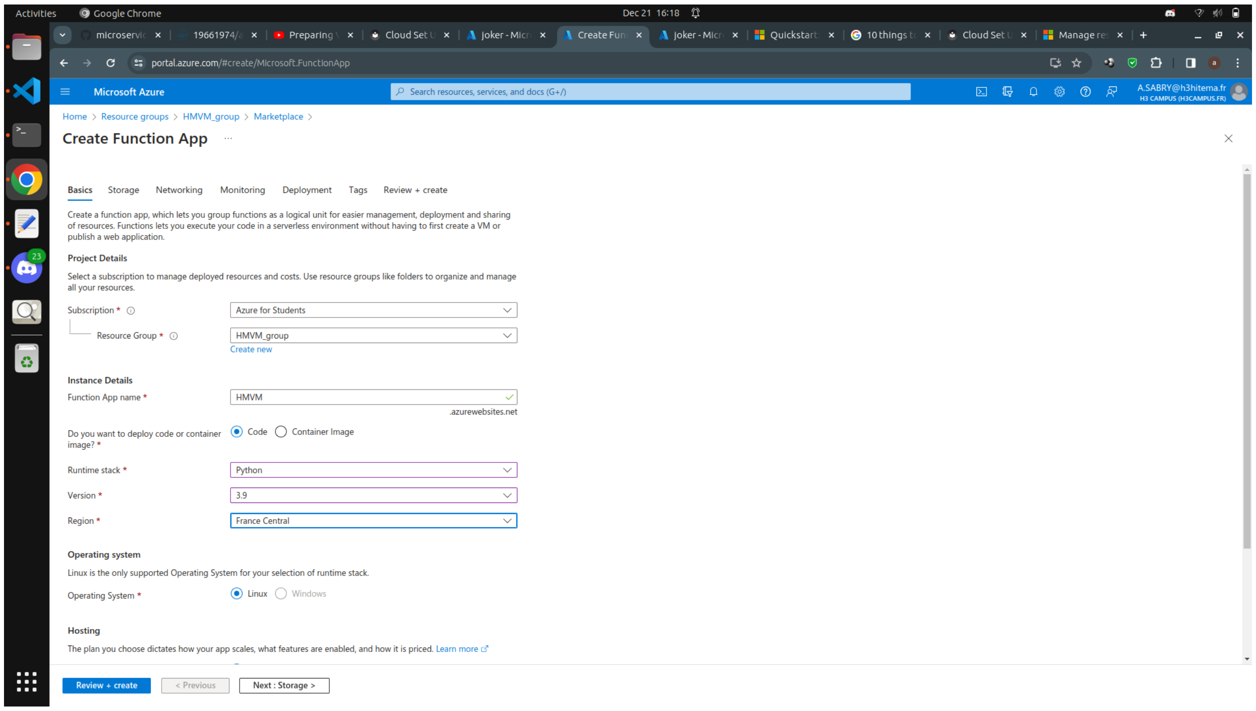
- Monitor the performance, metrics, and logs of your Storage Account using Azure Monitor or Storage Analytics.
- Manage settings, configure lifecycle management, and enable features like Azure Blob Versioning, Static Website hosting, etc.

Utilize Azure Storage Services:

- Use different services offered by Azure Storage such as Blob Storage for object storage, File Storage for file shares, Table Storage for NoSQL data, and Queue Storage for messaging.

Creating an Azure Storage Account enables you to store a wide range of data types and manage them securely and reliably within the Azure cloud ecosystem.

Steps to Create Functions in Python using Azure Functions:



Create Function App

Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: Azure for Students

Resource Group *: HMVM_group

Create new

Instance Details

Function App name *: HMVM .azurewebsites.net

Do you want to deploy code or container image? *: [✓ Code] Container Image

Runtime stack *: Python

Version *: 3.9

Region *: France Central

Operating system

Linux is the only supported Operating System for your selection of runtime stack.

Operating System *: [✓ Linux] Windows

Hosting

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. Learn more

Create a Function App:

- Click on "+ Create a resource" in the upper left-hand corner.
- Search for "Function App" and select it from the results.
- Click "Create" to start setting up your Function App.

Configure Function App Settings:

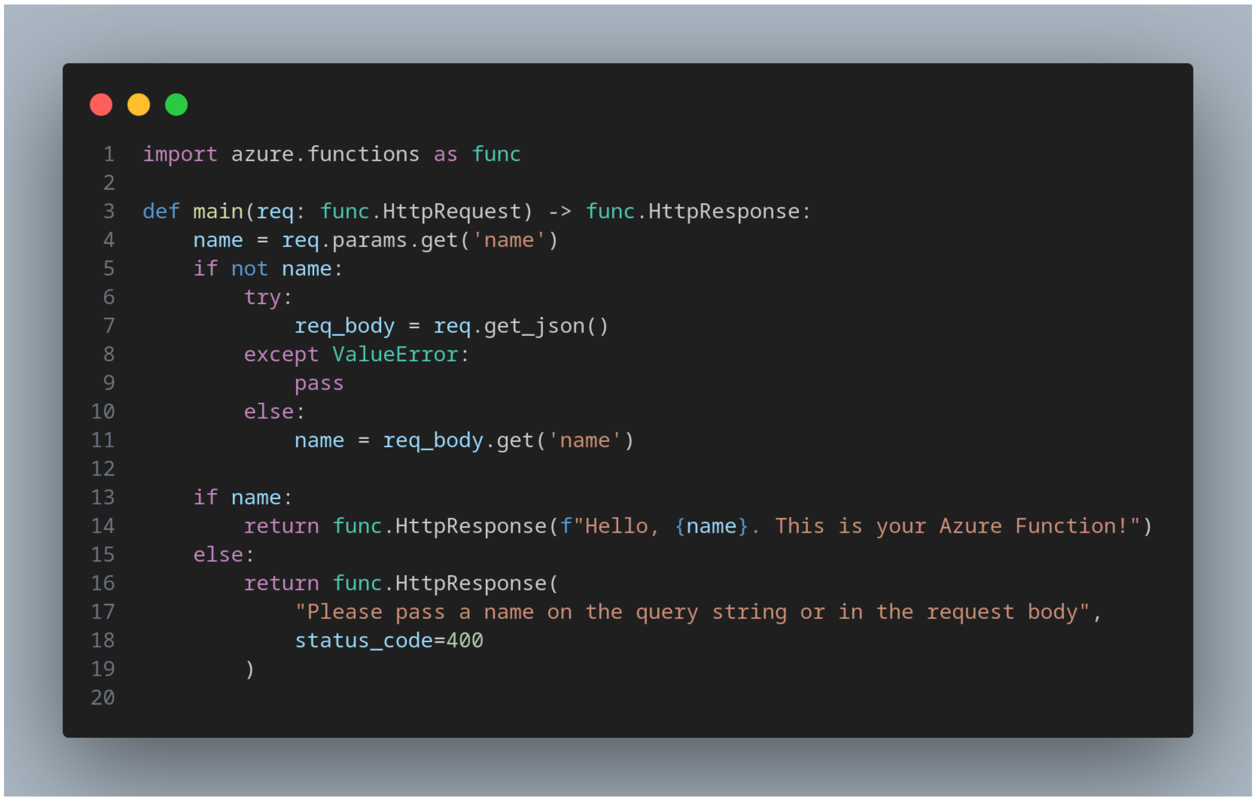
- Choose your Subscription and Resource Group.
- Provide a globally unique name for your Function App.
- Select the Runtime Stack as "Python".
- Choose the Operating System, Hosting Plan (consumption or app service plan), and Region.
- Set other options as needed, such as Application Insights for monitoring.

Create a Python Function within the Function App:

- Once the Function App is deployed, navigate to it in the Azure Portal.
- Click on the "+ Add" button to add a new function to your Function App.
- Choose a trigger for your function (HTTP trigger, Timer trigger, Blob trigger, etc.).
- Select Python as the language for your function.

Example (HTTP Trigger Function in Python):

Here's an example of an HTTP-triggered Azure Function using Python:



```python
import azure.functions as func

def main(req: func.HttpRequest) -> func.HttpResponse:
    name = req.params.get('name')
    if not name:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            name = req_body.get('name')

    if name:
        return func.HttpResponse(f"Hello, {name}. This is your Azure Function!")
    else:
        return func.HttpResponse(
            "Please pass a name on the query string or in the request body",
            status_code=400
        )

```

Test and Deploy the Function:

- Test your function within the Azure Portal using the provided testing tools.
- Once tested, deploy your Function App and its associated functions.

Monitor and Manage Functions:

- Monitor the performance and logs of your functions using Azure Monitor or Application Insights.
- Manage your functions by scaling, configuring triggers, setting up bindings, etc.

Integrate Functions into Your Workflow:

- Use the HTTP endpoint or trigger of your function within your applications or other Azure services.

Azure Functions in Python allow you to write serverless code effortlessly, triggering actions based on events or HTTP requests, providing a flexible and scalable solution for various scenarios.

Key Features and Steps for Azure VMware Solution (AVS):

Deploying VMware Workloads:

- Once the AVS is set up, you can create VMware virtual machines (VMs), migrate existing VMs, or deploy new VMs using vSphere Client or vCenter Server.
- Leverage VMware-compatible tools and interfaces to manage and operate your VMware workloads.

Integration with Azure Services:

- Benefit from Azure services by integrating with native Azure tools, such as Azure Backup, Azure Site Recovery, Azure Monitor, etc., for enhanced management, security, and monitoring of VMware workloads.

Networking and Security:

- Implement network configurations, security policies, and access controls using Azure services and VMware tools.
- Use Azure's security offerings like Azure Security Center for enhanced security posture.

Scalability and Performance:

- Scale your VMware environment by adding or removing hosts based on workload requirements.
- Utilize Azure's vast infrastructure to achieve high availability, performance, and scalability.

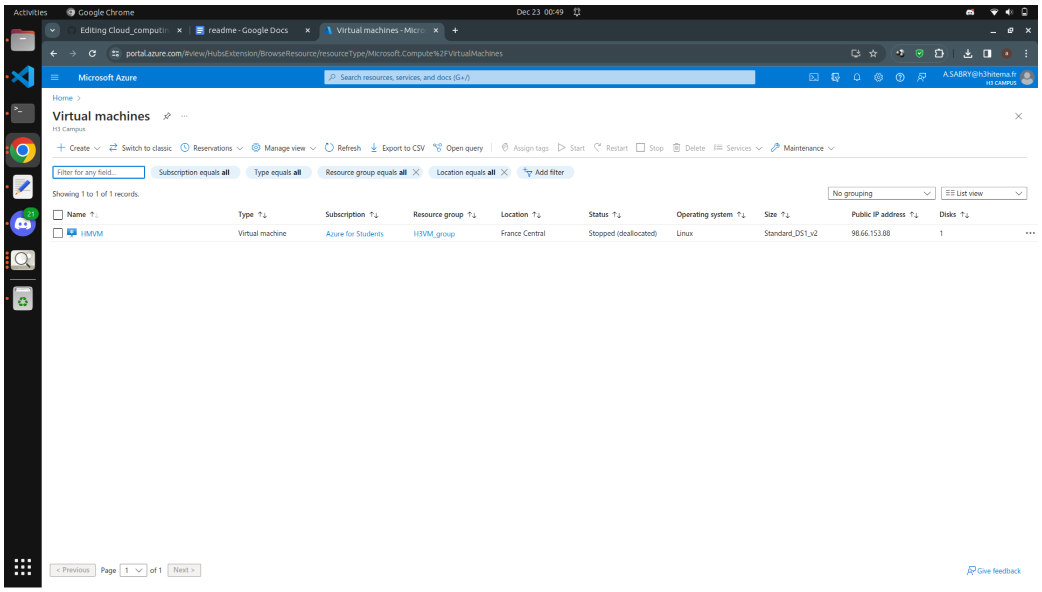
Cost Management and Optimization:

- Leverage Azure cost management tools to monitor and optimize spending on both Azure infrastructure and VMware workloads running in AVS.

Hybrid Cloud Scenarios:

- AVS enables seamless hybrid cloud scenarios, allowing you to extend your on-premises VMware environment to Azure, facilitating workload migration, disaster recovery, and data center extension.

Azure VMware Solution offers a fully managed service, providing a consistent operational experience across on-premises and Azure environments. It's designed for organizations seeking to leverage the benefits of both VMware technologies and Azure's global infrastructure, enabling a smoother transition to the cloud.


Virtual machines

H3 Campus

Showing 1 to 1 of 1 records.

Name	Type	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address	Disks
HMVM	Virtual machine	Azure for Students	H3VM_group	France Central	Stopped (deallocated)	Linux	Standard_DS1_v2	98.66.153.88	1

Steps to Work with Azure Blob Storage:

Access Blob Storage:

- Use Azure Portal, Azure CLI, Azure PowerShell, or SDKs (Python, .NET, Node.js, etc.) to interact with Blob Storage.
- Access storage account keys or Shared Access Signatures (SAS) to authorize access.

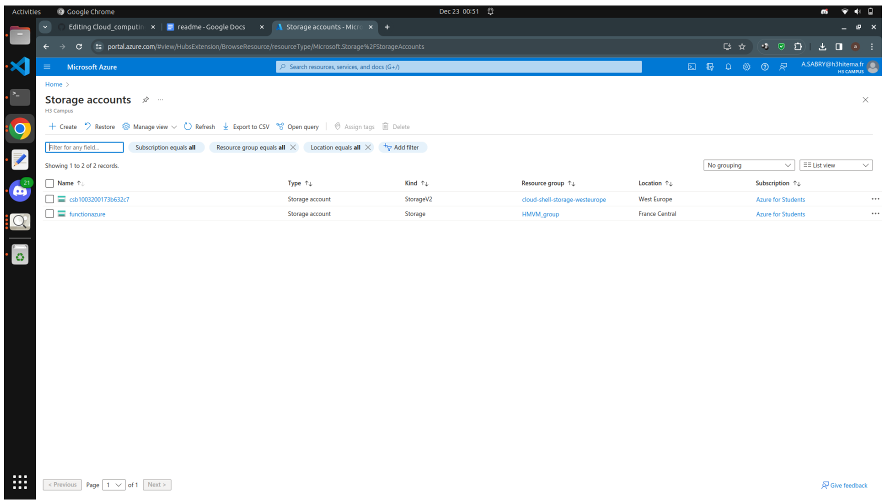
Create Containers and Upload Blobs:

- Create containers within your storage account to organize blobs.
- Upload blobs (files or data) into these containers using tools like Azure Storage Explorer, Azure CLI, or SDKs.

Manage Blob Lifecycle and Access:

- Set access policies, configure lifecycle management, and define storage tiers for optimal cost and performance.

Azure Blob Storage provides a highly scalable, secure, and cost-effective solution for storing and managing unstructured data, offering a variety of features to suit different storage needs within the Azure ecosystem.



Storage accounts

H3 Campus

Subscription equals **all** | Resource group equals **all** | Location equals **all**

Showing 1 to 2 of 2 records.

Name	Type	Kind	Resource group	Location	Subscription
csb1003200173b632c7	Storage account	StorageV2	cloud-shell-storage-westeurope	West Europe	Azure for Students
functionazure	Storage account	Storage	HMVM_group	France Central	Azure for Students

Steps to Deploy a Web App in Azure using Azure App Service:

Create an App Service Plan:

- Click on "+ Create a resource" in the upper left-hand corner.
- Search for "App Service Plan" and select it from the results.
- Click "Create" to set up your App Service Plan.
- Specify details like Subscription, Resource Group, Name, Region, and Pricing Tier (based on your app's needs).

Create a Web App:

- Once the App Service Plan is created, click on "+ Create a resource" again.
- Search for "Web App" and select it from the results.
- Provide details such as Subscription, Resource Group, Name, Runtime Stack (Node.js, .NET, Python, etc.), and Region.
- Connect the web app to the previously created App Service Plan.

Deploy Code to the Web App:

- After creating the Web App, you can deploy your code using different methods:
    - Deployment Center: Use Azure DevOps, GitHub, Bitbucket, FTP, or local Git to deploy code.
    - FTP/SFTP: Upload your web app files via FTP/SFTP using tools like FileZilla.
    - Azure CLI or Azure PowerShell: Use command-line interfaces to deploy code.

Configure Settings:

- Set up Application Settings, Connection Strings, environment variables, and configurations specific to your web app.
- Configure SSL certificates, custom domains, and other security-related settings.

Monitor and Scale:

- Monitor your web app's performance, logs, and metrics using Azure Monitor or Application Insights.
- Configure auto-scaling rules to manage resources based on demand.

Test Your Web App:

- Access your web app using the provided URL or custom domain to ensure it's functioning as expected.

Additional Features:

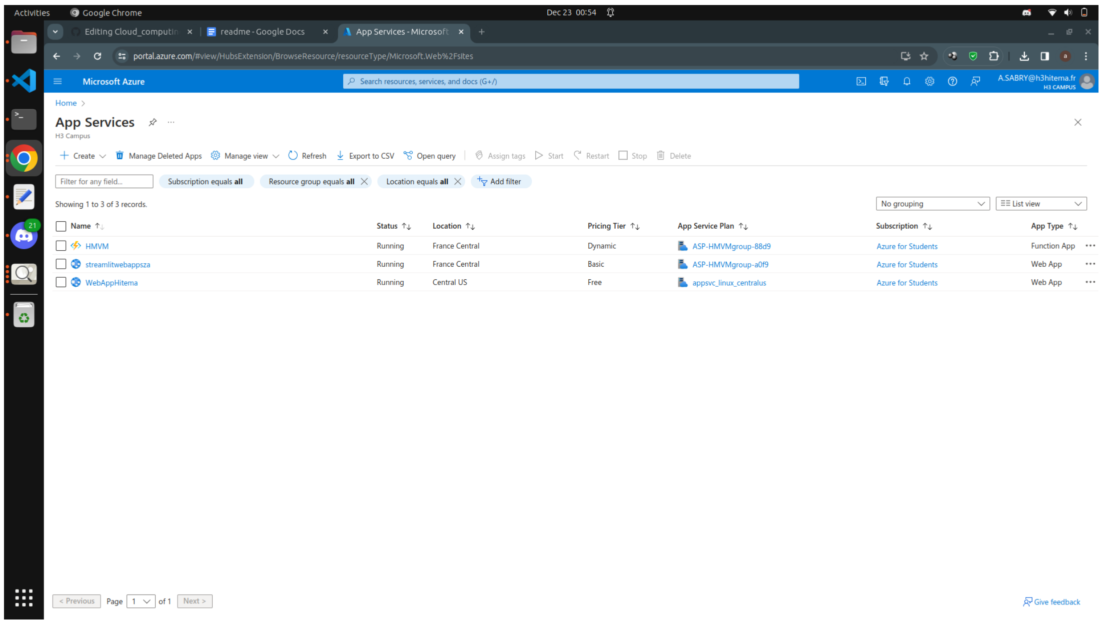
Continuous Deployment:

- Set up continuous integration/deployment pipelines for automated deployments when new code changes are pushed to repositories.. Backup and Restore:
- Configure backup and restore options to protect your web app's data and settings.

Integrate with Other Azure Services:

- Utilize Azure services like Azure SQL Database, Azure Blob Storage, Azure Functions, etc., for enhanced functionality within your web app.

Deploying a web app in Azure App Service provides a scalable and managed environment for hosting your applications, offering various deployment options and features to suit different development and deployment needs.



App Services

H3 Campus

Subscription equals **all** | Resource group equals **all** | Location equals **all** | Add filter

Showing 1 to 3 of 3 records.

Name	Status	Location	Pricing Tier	App Service Plan	Subscription	App Type
HMVM	Running	France Central	Dynamic	ASP-HMVMgroup-88d9	Azure for Students	Function App
streamlitwebappsza	Running	France Central	Basic	ASP-HMVMgroup-a0f9	Azure for Students	Web App
WebAppHitema	Running	Central US	Free	appsvc_linux_centralus	Azure for Students	Web App