Chani Refson
Task 4 - Theoretical part

1. Differences between object storage and other forms of distributed storage

	Object storage	**Block storage**	**Cloud file storage**
How doing	Object storage manages data as discrete, flat-structured units using unique IDs	Block storage divides data into fixed-sized, individually addressable units.	Cloud file storage provides shared, hierarchical access managed by providers.
Physical storage	Distributed across multiple storage nodes.	Distributed across SSDs and HDDs.	NAS servers run on-premises or over physical block storage.
Scalability	Scales infinitely to petabytes and beyond	Potentially scales up to millions of files, but can't handle more	Somewhat limited.
File Management	Requires APIs and new code for existing application access.	Supports files but needs more budget and management resources.	Supports standard protocols and works with existing shared applications.
MetaData Managment	Object storage enables unlimited and customizable metadata for any object.	Uses very little associated metadata.	Stores limited metadata relevant to files only.
Performence	Stores unlimited data with minimal latency.	High-performance, low latency, and rapid data transfer.	Offers high performance for shared file access.

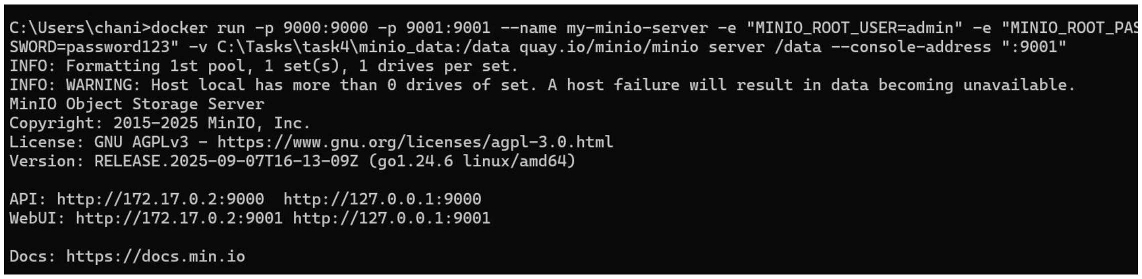
2. What is S3?
S3 is a scalable object storage service for diverse data use-cases.
    - S3 offers virtually unlimited, elastic storage with pay-per-use pricing
    - Amazon S3 delivers industry-leading 11 nines durability and 99.99% availability.
    - S3 ensures data protection through default encryption, security controls, and auditing.
    - S3 optimizes costs through multiple storage classes and automated lifecycle management.

3. An S3 bucket is a cloud container for object-based storage, housing data, unique IDs, and metadata instead of a traditional file system. S3 buckets provide infinite cloud storage for backups, data lakes, websites, and enterprise applications.
4. S3 uses a **flat structure**, not real folders. The "folder" concept is simulated using **name prefixes** (keys). For example, in `photos/myphoto.jpg`, the string `photos/` is just a prefix that the console displays as a folder for organizational convenience. S3 supports nested folders but not nested buckets; folders can be created or deleted, but never renamed.
5. No, Amazon S3 buckets (and their "folders") have no inherent size limits for total storage or number of objects; they scale elastically, but individual objects (files) have a max size of 5 TB (or 50 TB as of Dec 2025), requiring multipart uploads for anything over 100 MB or larger than 5GB/50TB for single uploads, using keys (like folder/file.txt) to create a folder hierarchy.

	S3	**Classic Filesystem**
Structure	Flat; "folders" are just part of the object's unique key (name), like myfolder/myfile.txt.	Hierarchical; actual directories with specific inode/metadata management.
Limits	Unlimited bucket size, up to 50 TB per object (using multipart upload).	Limited by physical disk size (though easily expandable).
Best For	Large files, backups, static content, web assets, data lakes.	Applications needing frequent, small, random reads/writes, databases, operating systems.
Behavior	Designed for putting and getting entire objects; partial writes are inefficient without caching layers.	Supports efficient partial file overwrites and random access.

6. S3 implementations fall into three main categories:
   - Public Cloud Services: Major providers offer S3-compatible storage for easy migration and cost-efficiency, including Google Cloud Storage, Cloudflare R2 (no egress fees), DigitalOcean Spaces, and budget-friendly specialists like Wasabi and Backblaze B2.
   - Open-Source & Self-Hosted: Software for building private S3 clouds on your own servers, such as MinIO (high performance), Ceph (enterprise-grade), SeaweedFS (optimized for small files), and Garage (distributed/flexible).
   - Hardware Appliances: Ready-to-use physical storage servers with built-in S3 API support from vendors like NetApp StorageGRID, Dell EMC ECS, and Cloudian HyperStore.

7.

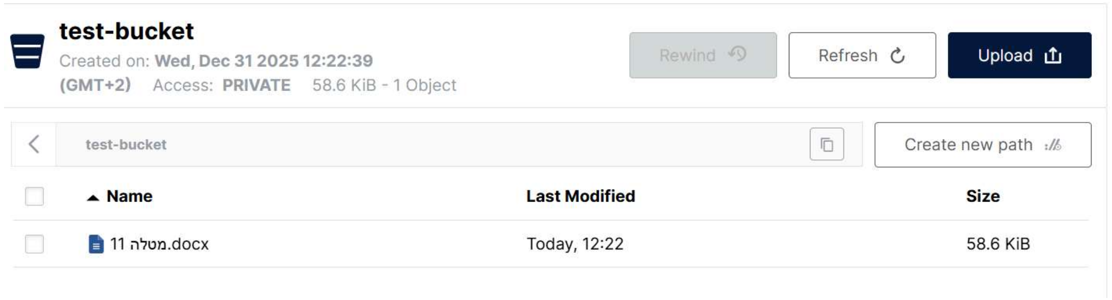


```
C:\Users\chani>docker run -p 9000:9000 -p 9001:9001 --name my-minio-server -e "MINIO_ROOT_USER=admin" -e "MINIO_ROOT_PAS
SWORD=password123" -v C:\Tasks\task4\minio_data:/data quay.io/minio/minio server /data --console-address ":9001"
INFO: Formatting 1st pool, 1 set(s), 1 drives per set.
INFO: WARNING: Host local has more than 0 drives of set. A host failure will result in data becoming unavailable.
MinIO Object Storage Server
Copyright: 2015-2025 MinIO, Inc.
License: GNU AGPLv3 - https://www.gnu.org/licenses/agpl-3.0.html
Version: RELEASE.2025-09-07T16-13-09Z (go1.24.6 linux/amd64)

API: http://172.17.0.2:9000  http://127.0.0.1:9000
WebUI: http://172.17.0.2:9001 http://127.0.0.1:9001

Docs: https://docs.min.io
```

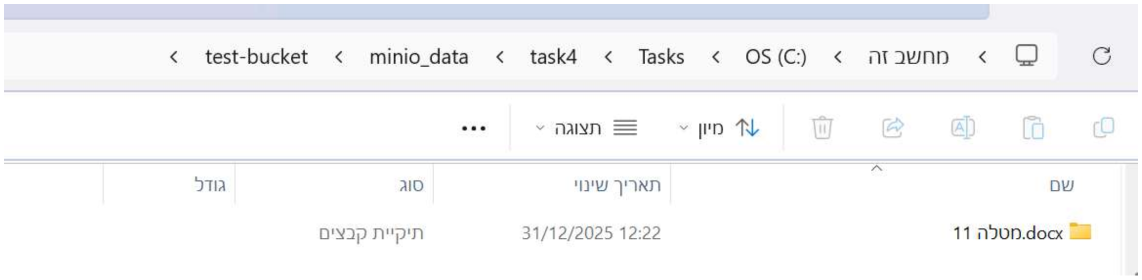
Figure 1: MinIO Server Successfully Deployed via Docker This screenshot shows the running container logs. It confirms that the MinIO server is active, with the API listening on port 9000 and the Web Console available on port 9001.



test-bucket
Created on: **Wed, Dec 31 2025 12:22:39 (GMT+2)** Access: **PRIVATE** 58.6 KiB - 1 Object

Name	Last Modified	Size
11 מטלה.docx	Today, 12:22	58.6 KiB

Figure 2: MinIO Console and Object Management This screenshot demonstrates the functional MinIO Web User Interface (Console). I successfully created a bucket named "test-bucket" and uploaded a document. The interface displays the object's metadata, including its size (58.6 KiB) and the upload timestamp, confirming that the storage service is fully operational.



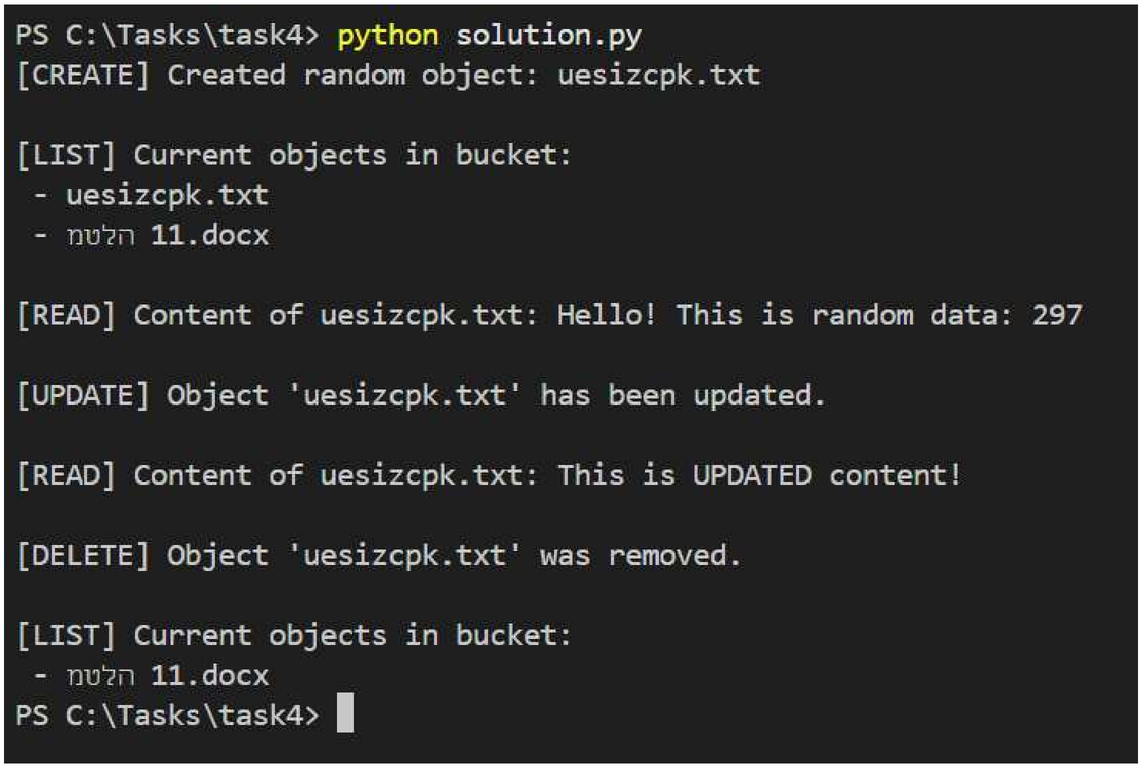
מחשב זה > OS (C:) > Tasks > task4 > minio_data > test-bucket
מיון
תצוגה

שם	תאריך שינוי	סוג	גודל
11 מטלה.docx	31/12/2025 12:22	תיקיית קבצים	

Figure 3: Host File System Integration (Data Persistence) This screenshot confirms that the Docker Volume Mapping is working correctly. The file uploaded via the MinIO Web Console is visible directly on the host machine's Windows File Explorer at C:\Tasks\task4\minio_data. This ensures that all data remains persistent even if the container is stopped or removed.

8. Python file of this answer exist in the folder.
The Role of Versioning in Object Updates: "In Object Storage, objects are immutable, meaning they cannot be partially edited. An update is technically a full overwrite. When Versioning is enabled, MinIO assigns a unique Version ID to every upload. Instead of losing the previous data during an update, the system retains the old version as a 'non-current' copy and sets the new upload as the 'latest' version. This provides data redundancy, allowing users to recover from accidental overwrites or deletes by reverting to a previous Version ID.

screenshots:



```
PS C:\Tasks\task4> python solution.py
[CREATE] Created random object: uesizcpk.txt

[LIST] Current objects in bucket:
 - uesizcpk.txt
 - מטלה 11.docx

[READ] Content of uesizcpk.txt: Hello! This is random data: 297

[UPDATE] Object 'uesizcpk.txt' has been updated.

[READ] Content of uesizcpk.txt: This is UPDATED content!

[DELETE] Object 'uesizcpk.txt' was removed.

[LIST] Current objects in bucket:
 - מטלה 11.docx
PS C:\Tasks\task4>
```