ICT-40-2020
H2020-ICT-2018-20

DataCloud

ENABLING THE BIG DATA PIPELINE LIFECYCLE ON THE COMPUTING CONTINUUM

D4.3: APPLICATIONS AND APPLIANCES REPOSITORY V2

https://datacloudproject.eu/

Document Metadata

Work package	WP4
Date	24.10.2022
Deliverable editor	Dragi Kimovski (AAU)
Version	1.0
Contributors	Zahra Najafabadi (AAU), Narges Mehran (AAU)
Reviewers	Ioannis Plakas (UBI), Fernando Perales (JOT)
Keywords	Docker Repository, Object-based repository
Dissemination Level	Public

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	10/01/2022	Structure of the deliverable	Dragi Kimovski (AAU)
V0.2	22/02/2022	Section 2	Narges Mehran (AAU)
V0.3	20/03/2022	Section 3	Narges Mehran (AAU)
V0.4	20/04/2022	Section 3	Zahra Najafabadi (AAU)
V0.5	28/05/2022	Section 3, 4	Zahra Najafabadi (AAU)
V0.6	20/08/2022	Section 2, 3	Zahra Najafabadi (AAU)
V0.7	30/09/2022	Revision of the first completed draft	Dragi Kimovski (AAU)
V0.8	15/10/2022	Reviewers’ comments addressed	Zahra Najafabadi (AAU)
V1.0	24/10/2022	Final formatting and layout	Brian Elvesæter (SI)

DataCloud Project Partners

SI	SINTEF AS
URO	SAPIENZA UNIVERSITY OF ROME
AAU	UNIVERSITY OF KLAGENFURT
KTH	ROYAL INSTITUTE OF TECHNOLOGY
IEX	IEXEC BLOCKCHAIN TECH
UBI	UBITECH
JOT	JOT INTERNET MEDIA
MOG	MOG TECHNOLOGIES SA
CER	CERAMICA CATALANO SRL

Co-funded by the Horizon 2020
Framework Programme of the European Union

TLE	TELLU IOT AS
BOS	ROBERT BOSCH GMBH

DISCLAIMER

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101016835.

Co-funded by the Horizon 2020
Framework Programme of the European Union

EXECUTIVE SUMMARY

Deliverable 4.3 explores currently available secure repository management systems for storing applications, models, and container templates for the pipelines developed in WP3 and provisioned on the Cloud resources. The deliverable provides information on the updates of the repository conducted in relation to the requirements of the DataCloud use-case partners. Therefore, in this deliverable, we provide information on the extension of the MinIO repository with support for storage, versioning, and management of Docker containers.

For the given purpose in this deliverable, we have:

1. Explored multiple approaches for the support of docker repositories.
2. Deployed a distributed storage repository at the University of Klagenfurt with support for docker containers.
3. Provided a detailed user manual for programmatic and interface-based access to the docker repository over the MinIO framework.

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF LISTINGS

Co-funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

MC	MinIO Client
MQTT	Message queuing telemetry transport
IoT	Internet of things

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

Deliverable 4.3 provides a detailed description of the updated storage repository with support for Docker images. We have considered the requirements of the use-case application and extended the classical MinIO object storage repository with functionality for management, versioning and deployment of Docker images. Therefore, in this section we provide background information on the MinIO storage system and Docker containers.

MinIO is software-defined high-performance object storage released under GNU Affero General Public License v3.0. Besides, we selected MinIO as it is open source and relatively straightforward to deploy through a Docker swarm and Kubernetes.

MinIO splits and replicates the data into “chunks” or parts, which helps to protect the data against failures such as corruption, hardware failure, or intrusions by using “Erasure code”. This tool’s feature helps achieve a high level of redundancy, and it is possible to lose up to half (N/2) of the total storage devices and still recover the data.

More information about MinIO can be found in Deliverable D4.1.

Docker is a platform for developers to build, ship, and run distributed applications with the help of Docker Engine. The snapshot from a container's runtime is portable and can be stored in Docker Hub as a cloud-based registry and code repository. The majority of use-case applications in DataCloud rely on Docker containers for application delivery, and therefore, it is essential to allow transparent management of Docker images. The Docker approach provides multiple advantages over classical virtual machine based deliver, including:

- Faster application delivery - Docker is appropriate in aiding the development lifecycle. While the developers are working in isolated operation teams inside a company, Docker technology allows them to develop containers consisting of separate (micro)services, integrate them, and deploy the whole workflow for their customers.
- Elastic deployment and scaling - Highly portable workloads packaged in Docker containers are able to be executed consistently on physical Edge devices or the Cloud virtual machines.

Co-funded by the Horizon 2020
Framework Programme of the European Union

2 BACKGROUND AND RELATED WORK

MinIO supports three deployment strategies, single-node single-drive, single-node multi-drive, multi-node multi-drive.

1. Single-node Single-drive (standalone): is suitable for a single MinIO server deployed on a volume of storage which is useful for evaluation and initial application development. This deployment provides a high level of reliability and fault tolerance. Standalone MinIO deployment provides a zero–parity for erasure coding, including the following dependent features:
    a. Versioning
    b. Object locking/ retention
2. Single-node Multi-drive (standalone multi-drive): supports a single MinIO server for several volumes of storage, providing high availability and redundancy. This deployment support erasure coding with the following dependent features:
    a. Object versioning
    b. Retention: write-once-read-many, object locking
3. Multi-node Multi-drive (distributed): supports several MinIO servers with multiple storage volumes.

In repository version two (v2), we extended the single-node single-drive deployment of the MinIO software suite to the single-node multiple-drive deployment, which increases the reliability and fault tolerance. Moreover, we applied the operations, administration, and development of MinIO deployments on Docker containers.

Previously, several projects utilized the MinIO storage software by the Docker containers in their tools. Poojara et al. [2] proposed a serverless data pipeline approach designed based on the Apache NiFi, MinIO, and MQTT services. Andersen et al. [3] designed a multi-national cluster based on the Kubernetes orchestration framework while utilizing a MinIO database for the images and videos processed by their surveillance management applications. Kennouche et al. [4] utilized MinIO to store the trained model on the storage system accessible through the Kubernetes cluster running on the OpenStack platform.

Therefore, the **Key Innovation Aspect** of the DataCloud repository in comparisons to the related solutions is the utilisation of object-based storage with support for Docker registry, specifically tailored for secure storage of Big Data pipelines.

Co-funded by the Horizon 2020
Framework Programme of the European Union

3 DOWNLOAD AND INSTALATION OF DOCKER ENGINE FOR MINIO

This section presents the different steps of extracting the MinIO software from the remote repository to our local machine.

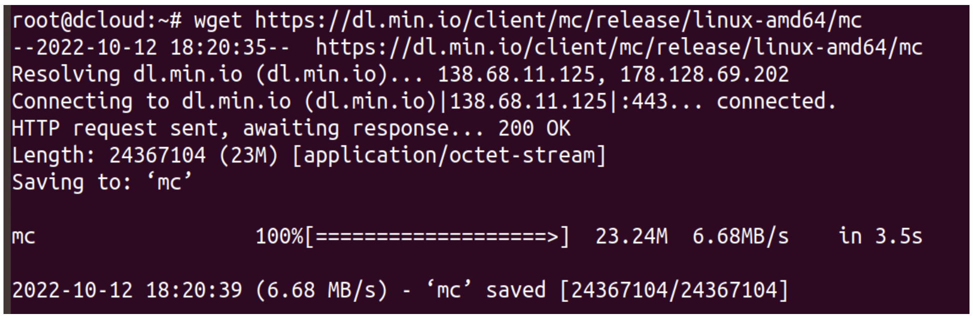
Firstly, we followed the instructions provided in Docker documentation [5] to install the Docker engine on our Ubuntu 20.04 server.

After the successful installation of the engine, we provided the steps followed to extract, install and set up the management dashboard for the MinIO admin.

3.1 INSTALATION OF THE MINIO CLIENT

To download the MinIO client executable on the local data center infrastructure we refer to the following command and related address:

- `wget https://dl.min.io/client/mc/release/linux-amd64/mc`



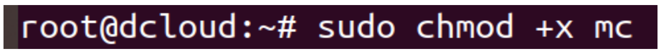
```
root@dcloud:~# wget https://dl.min.io/client/mc/release/linux-amd64/mc
--2022-10-12 18:20:35--  https://dl.min.io/client/mc/release/linux-amd64/mc
Resolving dl.min.io (dl.min.io)... 138.68.11.125, 178.128.69.202
Connecting to dl.min.io (dl.min.io)|138.68.11.125|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24367104 (23M) [application/octet-stream]
Saving to: ‘mc’

mc                   100%[===================>]  23.24M  6.68MB/s    in 3.5s

2022-10-12 18:20:39 (6.68 MB/s) - ‘mc’ saved [24367104/24367104]
```

Listing 1: The local installation procedure

To set the execution mode of the MinIO client application, we ran the following command:



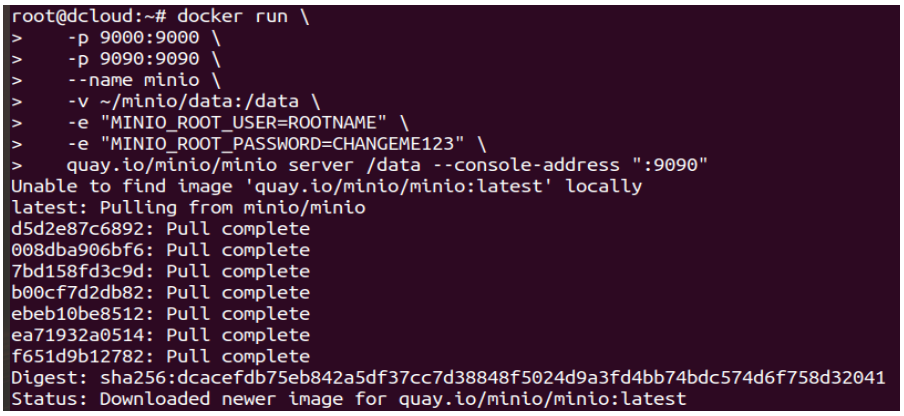
```
root@dcloud:~# sudo chmod +x mc
```

Listing 2: Command for setting up the execution mode

3.2 INSTALLATION THROUGH DOCKER ENGINE

We run the MinIO as a Docker container by setting the username, password, and port numbers. Before that, it is necessary to download the Docker image (`quay.io/minio/minio`) if it is not available in the repository:

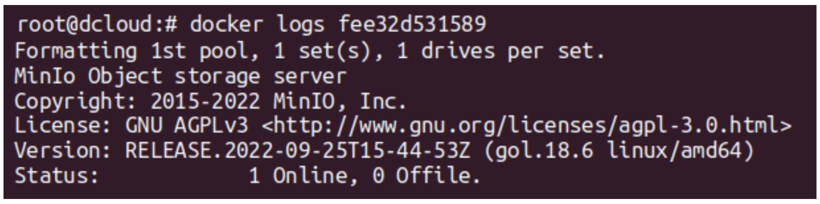
Co-funded by the Horizon 2020
Framework Programme of the European Union



```
root@dcloud:~# docker run \
>     -p 9000:9000 \
>     -p 9090:9090 \
>     --name minio \
>     -v ~/minio/data:/data \
>     -e "MINIO_ROOT_USER=ROOTNAME" \
>     -e "MINIO_ROOT_PASSWORD=CHANGEME123" \
>     quay.io/minio/minio server /data --console-address ":9090"
Unable to find image 'quay.io/minio/minio:latest' locally
latest: Pulling from minio/minio
d5d2e87c6892: Pull complete
008dba906bf6: Pull complete
7bd158fd3c9d: Pull complete
b00cf7d2db82: Pull complete
ebeb10be8512: Pull complete
ea71932a0514: Pull complete
f651d9b12782: Pull complete
Digest: sha256:dcacefdb75eb842a5df37cc7d38848f5024d9a3fd4bb74bdc574d6f758d32041
Status: Downloaded newer image for quay.io/minio/minio:latest
```

Listing 3: Running the MinIO Docker container on a Docker engine

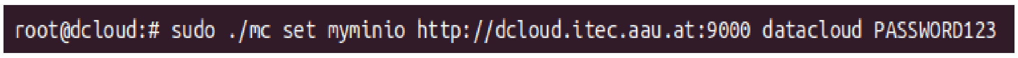
Besides, these are the logs of the running MinIO Docker container (ID = fee32d531589):



```
root@dcloud:# docker logs fee32d531589
Formatting 1st pool, 1 set(s), 1 drives per set.
MinIo Object storage server
Copyright: 2015-2022 MinIO, Inc.
License: GNU AGPLv3 <http://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2022-09-25T15-44-53Z (gol.18.6 linux/amd64)
Status:           1 Online, 0 Offile.
```

Listing 4: Logs of MinIO Docker execution

Then, we set the MinIO client to access the dashboard through the web browser:



```
root@dcloud:# sudo ./mc set myminio http://dcloud.itec.aau.at:9000 datacloud PASSWORD123
```

Listing 5: Configuring access from a web interface

Figure 1 shows the login page for admins and service users (the consortium). Afterward, it is possible to browse the dashboard of the application to manage the object storage service.

Co-funded by the Horizon 2020
Framework Programme of the European Union

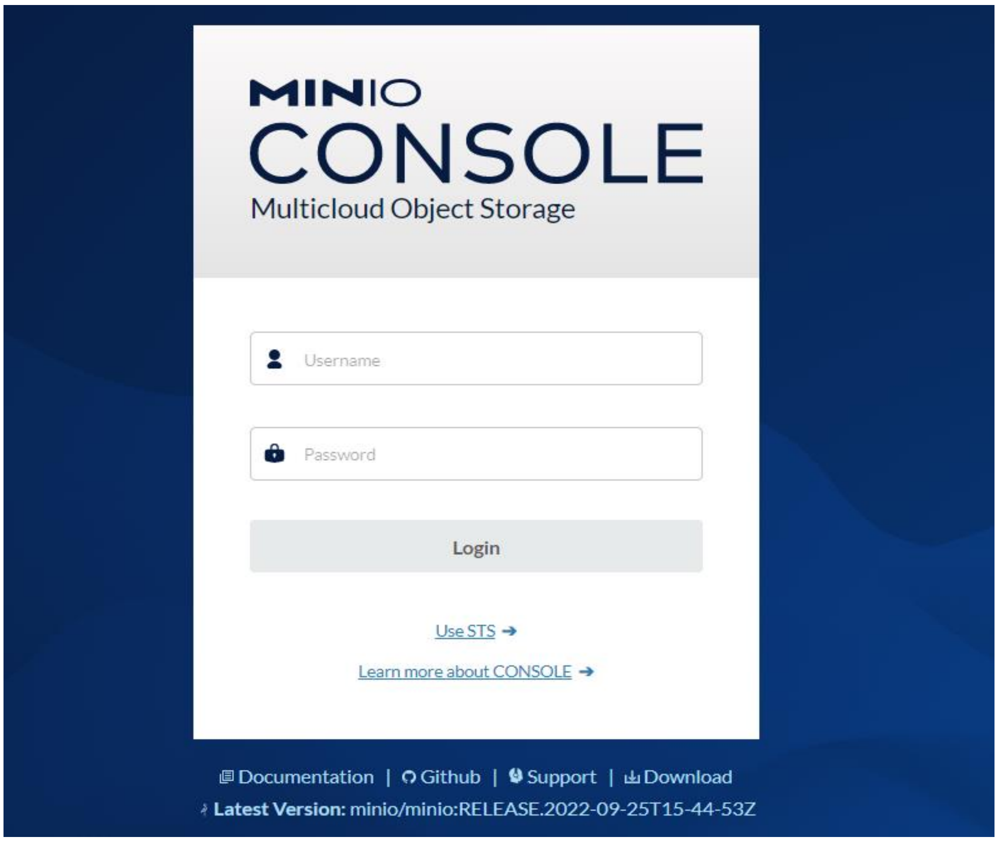

MINIO
CONSOLE
Multicloud Object Storage
Username
Password
Login
Use STS
Learn more about CONSOLE
Documentation | Github | Support | Download
Latest Version: minio/minio:RELEASE.2022-09-25T15-44-53Z

Figure 1: MinIO Login Console

Single-node multi-drive deployment: as observed in Figure 2, we used the MinIO console for the administration of a single-node multi-drive storage, such as Identity and access management, metrics and log monitoring, or server configuration. The figure shows four storage drives, each with a capacity of `933GB`. All these storage drives are managed by the admin team to be split among the consortium partners.

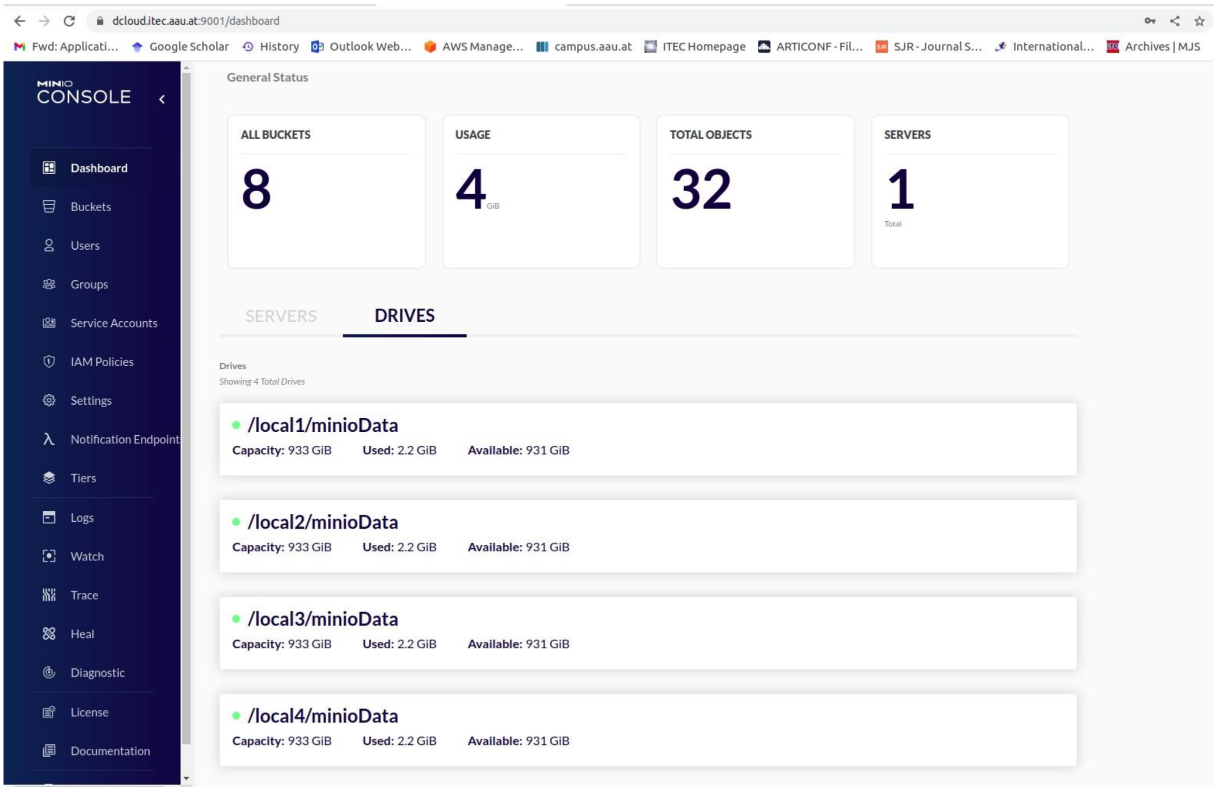
Moreover, it is possible to use the MinIO Client (mc) command line tool, providing a modern alternative to UNIX commands like ls, cat, cp, mirror, and diff. The **mc** command-line tool is built for compatibility with the AWS S3 API and is tested MinIO and AWS S3 for expected functionality and behaviour. mc has the following syntax:

- `mc [GLOBALFLAGS] COMMAND`

In addition, we can use the `mc admin info` command to test the connection to the newly added MinIO deployment:

- `mc admin info myminio`

Co-funded by the Horizon 2020 Framework Programme of the European Union



General Status

ALL BUCKETS 8

USAGE 4 GiB

TOTAL OBJECTS 32

SERVERS 1 Total

Drives
Showing 4 Total Drives

/local1/minioData
Capacity: 933 GiB **Used:** 2.2 GiB **Available:** 931 GiB

/local2/minioData
Capacity: 933 GiB **Used:** 2.2 GiB **Available:** 931 GiB

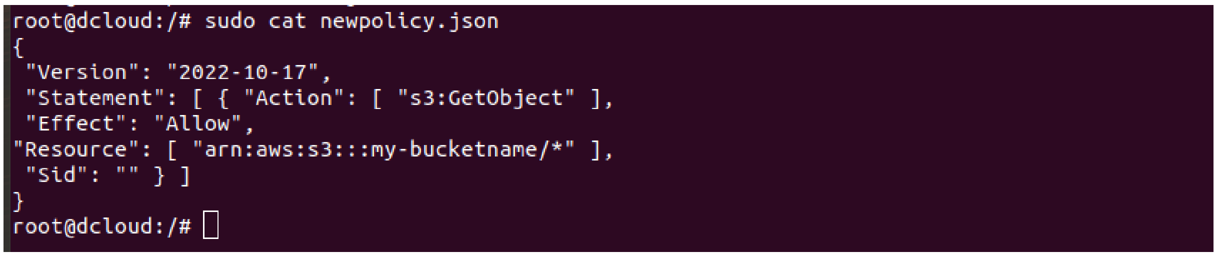
/local3/minioData
Capacity: 933 GiB **Used:** 2.2 GiB **Available:** 931 GiB

/local4/minioData
Capacity: 933 GiB **Used:** 2.2 GiB **Available:** 931 GiB

Figure 2: Storage drives available in the dashboard.

Minio access Policy: manages users accessibility *to the* Minio object storage. We created two buckets for each partner, where each bucket compromises several directories. The first bucket is private and accessible only by the specific partner, while the second is shared with all partners. To this end, we created a policy for each bucket and specified its accessibility through the following steps:

- Write a policy in json file through the nano command line tool (nano newpolicy.json):



```
root@dcloud:/# sudo cat newpolicy.json
{
 "Version": "2022-10-17",
 "Statement": [ { "Action": [ "s3:GetObject" ],
 "Effect": "Allow",
"Resource": [ "arn:aws:s3:::my-bucketname/*" ],
 "Sid": "" } ]
}
root@dcloud:/# 
```

Listing 6: Configuring a json file for user access policy at the MinIO repository

The "Action" specify the user action, while "Resource" indicates the bucket and the area that the user can access. my-bucketname is the name of the bucket that the user can access if we put * instead of the name of the particular bucket means that the user can access all the buckets in the Minio. Moreover, if we want to give the write and read right to the user, we can write * in front of the action.

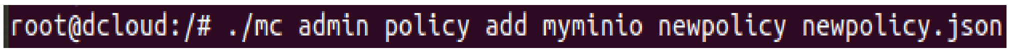
Notice: if we want to give access of several bucket to the user, we add name of several bucket

"arn:aws:s3:::my-bucketname1/*",

"arn:aws:s3:::my-bucketname2/*"

Co-funded by the Horizon 2020
Framework Programme of the European Union

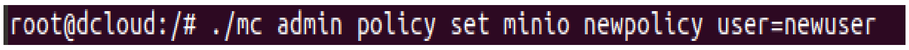
- Add the new policy to MinIO policy list through MinIO client (mc):



```
root@dcloud:/# ./mc admin policy add myminio newpolicy newpolicy.json
```

Listing 7: Adding new policy to minio policy list

- Attach the policy to the user through MinIO client:



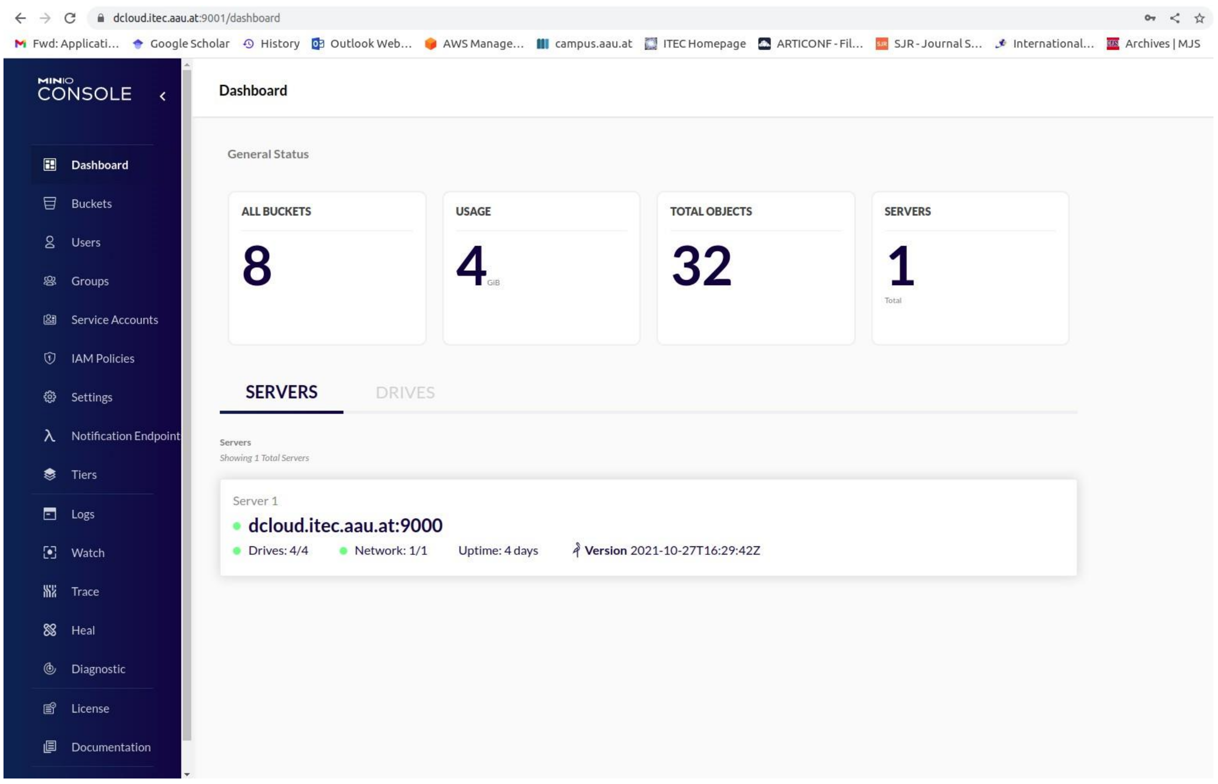
```
root@dcloud:/# ./mc admin policy set minio newpolicy user=newuser
```

Listing 8: Setting new policy to a minio user

Co-funded by the Horizon 2020
Framework Programme of the European Union

4 MINIO STORAGE USAGE INFORMATION

Figure 3 depicts that the administration dashboard provides a few monitoring metrics that can be observed, including the number of buckets created or the objects stored on the server machine:



Dashboard

General Status

ALL BUCKETS 8

USAGE 4 GiB

TOTAL OBJECTS 32

SERVERS 1 Total

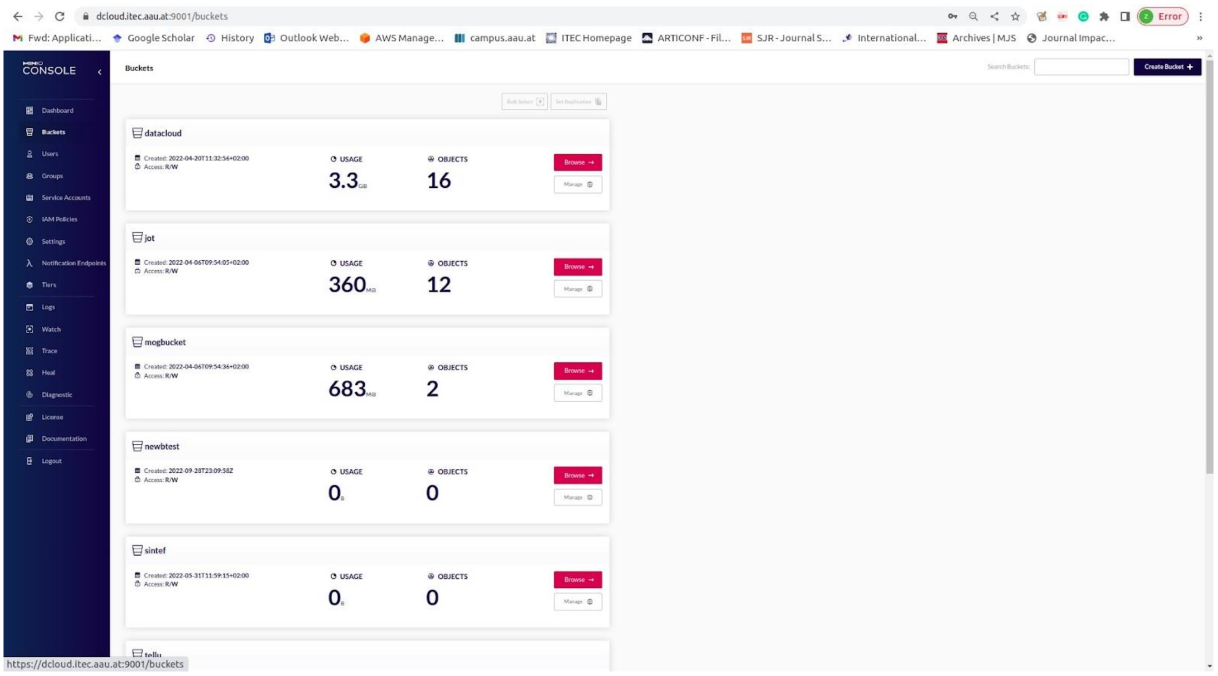
Servers
Showing 1 Total Servers

Server 1
dcloud.itec.aau.at:9000
Drives: 4/4 Network: 1/1 Uptime: 4 days Version 2021-10-27T16:29:42Z

Figure 3: General status for the admin user

The administrator is able to monitor the amount of storage used by the data of each project partner, along with the last access time to the system. As this is a few terabytes of storage, currently, there is no overutilization of resources (as observed in Figure 4). However, the management team will devise a plan in the case of high accessibility and usage of the object storage system.

Co-funded by the Horizon 2020 Framework Programme of the European Union



Buckets

datacloud
Created: 2022-04-20T11:32:56+02:00
Access: R/W
USAGE 3.3 GiB
OBJECTS 16

jot
Created: 2022-04-06T09:54:05+02:00
Access: R/W
USAGE 360 MiB
OBJECTS 12

mogbucket
Created: 2022-04-06T09:54:36+02:00
Access: R/W
USAGE 683 MiB
OBJECTS 2

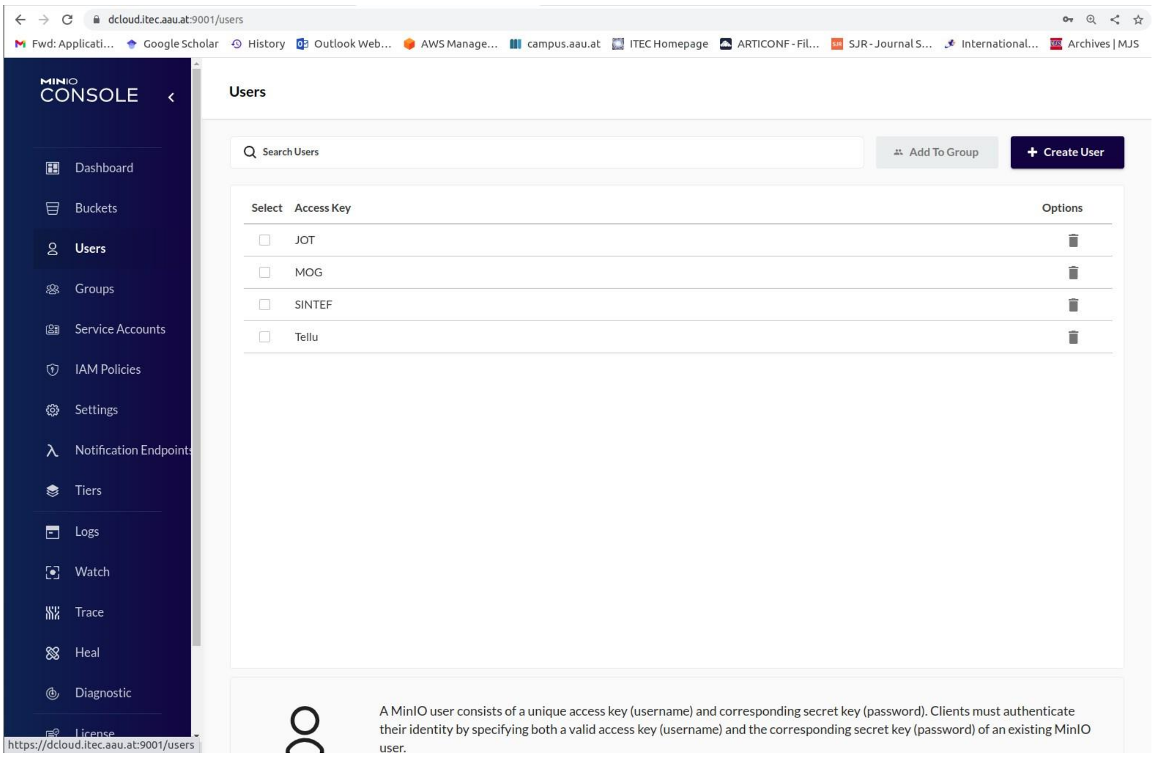
newbtest
Created: 2022-09-28T23:09:58Z
Access: R/W
USAGE 0 B
OBJECTS 0

sintef
Created: 2022-05-31T11:59:15+02:00
Access: R/W
USAGE 0 B
OBJECTS 0

tellu

Figure 4: Storage usage by the use-cases

Figure 5 shows that four partners currently set their credentials from the consortium and had access to their object storage management profiles.



Users

Select	Access Key	Options
☐	JOT	
☐	MOG	
☐	SINTEF	
☐	Tellu	

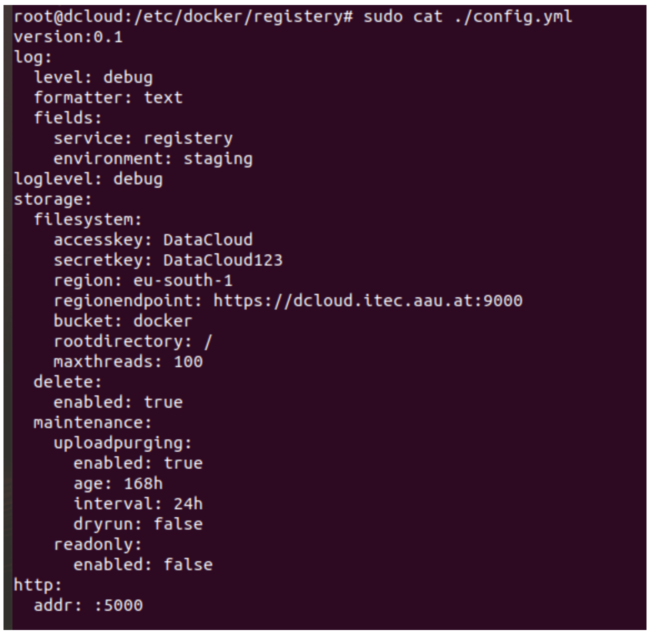
A MinIO user consists of a unique access key (username) and corresponding secret key (password). Clients must authenticate their identity by specifying both a valid access key (username) and the corresponding secret key (password) of an existing MinIO user.

Figure 5: Users accessing the distributed storage system

Co-funded by the Horizon 2020 Framework Programme of the European Union

5 CREATING DOCKER REGISTRY WITH MINIO

To create a simple private Docker registry, there exists a Docker image named "`registry:2`[1]" [7]. Firstly, we created the file `config.yml` in the directory of `/etc/docker/registry/` through the `nano` command line tool, and replaced the `accesskey`, `secretkey`, etc. with our setups.

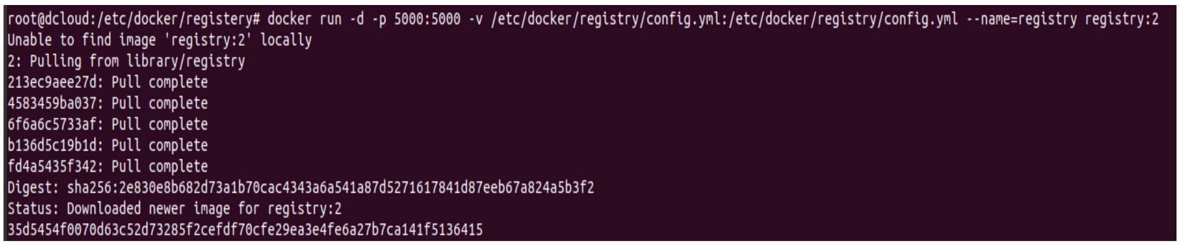


```
root@dcloud:/etc/docker/registery# sudo cat ./config.yml
version:0.1
log:
  level: debug
  formatter: text
  fields:
    service: registery
    environment: staging
loglevel: debug
storage:
  filesystem:
    accesskey: DataCloud
    secretkey: DataCloud123
    region: eu-south-1
    regionendpoint: https://dcloud.itec.aau.at:9000
    bucket: docker
    rootdirectory: /
    maxthreads: 100
  delete:
    enabled: true
  maintenance:
    uploadpurging:
      enabled: true
      age: 168h
      interval: 24h
      dryrun: false
    readonly:
      enabled: false
http:
  addr: :5000
```

Listing 9: Configuring a yaml file for Docker registry at the MinIO repository

After tagging the Docker image to dcloud.itec.aau.at:5000, we ran the following command to set up the registry based on the rules in the `config.yml` file with MinIO [8]:

- `docker run -d -p 5000:5000 -v /etc/docker/registry/config.yml:/etc/docker/registry/config.yml --name=registry registry:2`



```
root@dcloud:/etc/docker/registery# docker run -d -p 5000:5000 -v /etc/docker/registry/config.yml:/etc/docker/registry/config.yml --name=registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
213ec9aee27d: Pull complete
4583459ba037: Pull complete
6f6a6c5733af: Pull complete
b136d5c19b1d: Pull complete
fd4a5435f342: Pull complete
Digest: sha256:2e830e8b682d73a1b70cac4343a6a541a87d5271617841d87eeb67a824a5b3f2
Status: Downloaded newer image for registry:2
35d5454f0070d63c52d73285f2cefdf70cfe29ea3e4fe6a27b7ca141f5136415
```

Listing 10: Running Docker registry container with MinIO

Thereafter, we pushed the Docker tagged image into the already-created registry through the following command:

[1]https://hub.docker.com/_/registry

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
root@dcloud:# docker tag alpine:3.10 dcloud.itec.aau.at:5000/alpine:3.10
root@dcloud:# root@dcloud:/# docker push dcloud.itec.aau.at:5000/alpine:3.10
The push refers to repository [local host:5000/alpine]
35d5454f0070: Pushed
3.10: digest: sha256:451eee8bedcb2f029756dc3e9d73bab0e7943c1ac55cff3a4861c52a0fdd3e98 size: 528
```

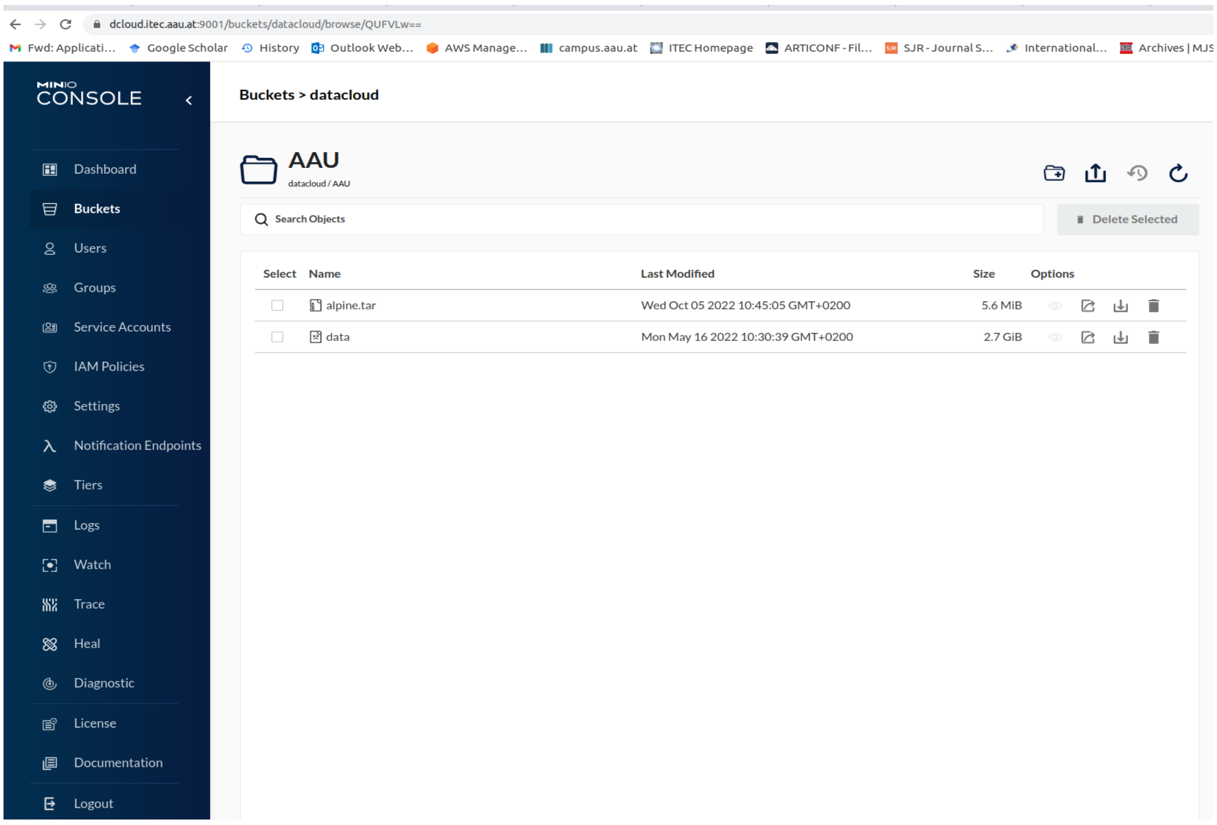
Listing 11: Tagging and pushing a Docker image to the registry

In addition, because a business case partner may need to provide one's software as a tarball file for the consortium, it is also possible to store every Docker image of a business case partner as the tarball, such as the 5.6MB `Alpine Linux v3.10` file:

```
root@dcloud:# docker save alpine:3.10 > alpine.tar
root@dcloud:# root@dcloud:/# ls -sh alpine.tar
5,6M alpine.tar
```

Listing 12: Saving a Docker image as a tarball to be stored in the MinIO repository

Afterward, Figure 6 shows that AAU has already uploaded the Docker image (`alpine.tar`) through the AAU console to the MinIO shared directory for `DataCloud` partners' access.



Select	Name	Last Modified	Size	Options
☐	alpine.tar	Wed Oct 05 2022 10:45:05 GMT+0200	5.6 MiB	
☐	data	Mon May 16 2022 10:30:39 GMT+0200	2.7 GiB	

Figure 6: Docker image shared for the consortium

Co-funded by the Horizon 2020
Framework Programme of the European Union

6 CONCLUSIONS

In this deliverable, we provided a detailed description on the extension of the functionalities of the available DataCloud distributed storage repository.

Based on the analysis we selected most appropriate repository management system that meets the requirements of DataCloud and extended it to support registration of Docker images. Thereafter, we selected proper storage hardware resources and integrated them as a stand-alone system in the Cloud data centre at the University of Klagenfurt, which allows transparent deployment of Docker images.

We provided external access to the storage system with possibility for both programmatic and UI-based access to all use case partners and presented usage statistics. Besides, we provide a user manual how to locally deploy MinIO with Docker image registry and how the stakeholders can properly use it.

Co-funded by the Horizon 2020
Framework Programme of the European Union

REFERENCES

[1] Minio. Kubernetes Native, High-Performance Object Storage, <https://min.io/> (2022).

[2] Poojara, S. R., Dehury, C. K., Jakovits, P., & Srirama, S. N. (2022). Serverless data pipeline approaches for IoT data in fog and cloud computing. Future Generation Computer Systems, 130, 91-105.

[3] Andersen, Hallvard Munkås, et al. "NATO Federated Coalition Cloud with Kubernetes: A National Prototype Perspective." (2021).

[4] Kennouche, Takai Eddine "Design of the localization & analytics as a service solution," PROJECT "LOCUS": localization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applications. 11/05/2021, Available at: <https://www.locus-project.eu/wp-content/uploads/2021/05/Deliverables-officially-submitted_D5.3_D5.3_11-5-21.pdf>

[5] How to install Docker engine. Available at: <https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>

[6] Docker-based installation of MinIOs. Available at: <https://docs.min.io/docs/minio-docker-quickstart-guide.html>

[7] Deploy a registry server for container content. Available at: <https://docs.docker.com/registry/deploying/>

[8] Using Minio with Docker Registry. Available at: <https://bweston.me/docker-registry-using-minio-the-ultimate-5675a53abcd0>

Co-funded by the Horizon 2020
Framework Programme of the European Union