Setup MinIO

Task Requirement

To set up an object storage system in a cluster using MinIO.

Environment Details

- **Operating System (OS):** Ubuntu 20.04
- **CPU:** 4
- **Storage:** 8 GB

List of Tools and Technologies

- **MinIO** - Latest Version (RELEASE.2023-08-16T20-17-30Z)
- **Docker** - Version 24.0.6
- **Sidekick**

Definitions of Tools

- **MinIO**: MinIO is a software-defined high-performance distributed object storage server. It can run on consumer or enterprise-grade hardware and supports various operating systems and architectures.
- **Docker**: Docker is a containerization platform that provides a consistent environment for running applications, making it easy to move and deploy applications across different systems.
- **Sidekick**: Sidekick is a high-performance sidecar load-balancer that eliminates centralized load balancer bottlenecks and simplifies DNS failover management by attaching a tiny load balancer to client application processes.

Commands for Setup or Configuration

1. Install Docker on Linux:

Update Package Lists:

Step 1

```
sudo apt update
```

`sudo:` This stands for "Superuser Do" and it's used to execute commands with elevated privileges. It allows you to make changes to your system that regular users wouldn't have permission to do.

`apt:` This is the package manager for Debian-based systems. It's used for handling packages—installing, updating, and removing them.

`update:` This is a specific command for apt. When you run sudo apt update, it doesn't actually update the packages on your system; instead, it updates the local database of available packages. This database is necessary for the package manager to know what packages are available, where to download them, and what versions exist.

Step 2

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

`sudo:` This is a command that allows a permitted user to execute a command as the superuser (or another user), as specified by the security policy. In other words, it allows you to run commands with administrative privileges.

`apt:` This is the package management command-line tool for Debian-based systems. It's used for installing, updating, upgrading, and removing software packages.

`install:` This is an argument for the apt command, indicating that you want to install one or more packages.

`apt-transport-https:` This package provides the "https" method for APT to access repositories securely over HTTPS. It's necessary for fetching packages from repositories that use HTTPS.

`ca-certificates:` This package contains trusted certificate authorities (CAs) certificates. These certificates are used to verify the authenticity of SSL/TLS connections, including those used when fetching packages over HTTPS.

`curl:` This is a command-line tool for making HTTP requests. It's often used for downloading files or interacting with web services. In this context, it's likely being installed to facilitate the secure download of packages and repository information.

`software-properties-common:` This package provides a common infrastructure for managing software repositories. It includes tools and utilities for adding and managing software repositories on the system.

Step 3

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

`curl:` This is a command-line tool for making HTTP requests. In this case, it's used to download content from a URL.

`-fsSL:` These are options for curl:

-f or --fail: Return an error status if the HTTP response code indicates an error. This is used to make sure that if the download fails, it doesn't continue with the rest of the command.

-s or --silent: Silent mode. It prevents curl from showing progress information or error messages.
-S or --show-error: Show error messages. This is used in combination with -s to show errors if they occur.

`https://download.docker.com/linux/ubuntu/gpg:` This is the URL from which curl is downloading content. In this case, it's downloading the GPG (GNU Privacy Guard) key for the Docker repository.

`|:` This is a pipe. It takes the output (in this case, the GPG key downloaded by curl) and passes it as input to the next command.

`sudo apt-key add -:` This command adds the GPG key to the list of trusted keys used by APT (Advanced Package Tool), the package management system used by Debian-based systems like Ubuntu. The - at the end of the command tells apt-key to read the key from the standard input, which is where the output of curl is being passed through the pipe.

Step 4

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

`sudo` : Executes the command with superuser privileges.

`add-apt-repository` : Adds a new repository to the system.

`"deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"` : The Docker repository information specifying the release (focal), architecture (amd64), and stability (stable).

Step 5

```
apt-cache policy docker-ce
```

`apt-cache:` This command is used to query package information from APT's package cache.

`policy:` This sub-command is used to display priority and origin information about packages, including installed versions and available versions.

`docker-ce:` This is the package name for Docker Community Edition. It's a specific version or edition of the Docker software.

Step 6

```
sudo apt install docker-ce
```

`sudo:` This is a command used in Unix-like operating systems to run a command with superuser privileges (root). It stands for "superuser do." In this context, it allows the user to install software, which typically requires elevated permissions.

`apt:` This is the package management command-line tool used in Debian-based systems (such as Ubuntu). It is used for handling packages, which are software packages containing precompiled binaries, configuration files, and other resources needed for the software to run.

`install:` This is the sub-command of apt used to install new packages.

`docker-ce:` This is the package name of Docker Community Edition (CE). Docker is a platform for developing, shipping, and running applications in containers. The docker-ce package includes the Docker engine, which is responsible for running containers.

2. Install Docker Compose:

Download Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)
```

`sudo:` This is a command in Unix-like operating systems that allows a permitted user to execute a command as the superuser or another user, as specified by the security policy.

`curl:` This command is used to transfer data to or from a server. In this case, it's used to download a file from the specified URL.

`-L:` This option tells curl to follow redirects. If the requested page has moved to a different location, curl will automatically follow the redirection to the new location.

`"https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)":` This is the URL from which curl is downloading a file. The URL contains placeholders $(uname - s)and$(uname -m), which are substituted with the system name and machine architecture using the uname command. This helps in getting the correct version of Docker Compose for the user's system.

`-o /usr/local/bin/docker-compose:` This option specifies the output file path and name for the downloaded file. In this case, docker-compose will be downloaded to the /usr/local/bin/ directory.

Make Docker Compose Executable:

```
sudo chmod +x /usr/local/bin/docker-compose
```

`sudo:` This command is used to execute another command with elevated privileges (usually as the superuser or root).

`chmod:` This command is used to change the permissions of a file or directory.

`+x:` This option for chmod stands for "execute" permission. It allows the file to be executed as a program.

`/usr/local/bin/docker-compose:` This is the path to the file for which we are changing permissions. In this case, it's the docker-compose executable located in the /usr/local/bin/ directory.

Verify Docker Compose Installation:

```
docker-compose --version
```

`docker-compose:` This is the executable for Docker Compose, a tool for defining and running multi-container Docker applications. It allows you to define your application's services, networks, and volumes in a docker-compose.yml file, and then start and run your entire application with a single command.

`--version:` This is a command-line option that instructs Docker Compose to display information about its version. When you run this command, Docker Compose will output the version number installed on your system.

3. Edit the `docker-compose.yml` File

Use the `vim` command to open and edit the `docker-compose.yml` file, which is commonly used for defining multi-container Docker applications.

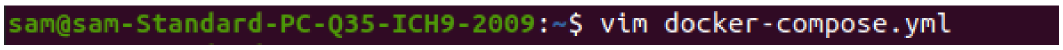
```
vim docker-compose.yml
```

Here's a breakdown of what this command does:

- **`vim`** : This is the command to launch the Vim text editor.
- **`docker-compose.yml`** : This is the filename you want to open and edit. In this case, it appears to be a Docker Compose configuration file, which is commonly used for defining multi-container Docker applications.

Once you run this command, Vim will open the `docker-compose.yml` file, allowing you to view and make changes to its contents. Vim has its own set of commands and shortcuts for navigating and editing text. If you're not familiar with Vim, here are some basic commands to get you started:

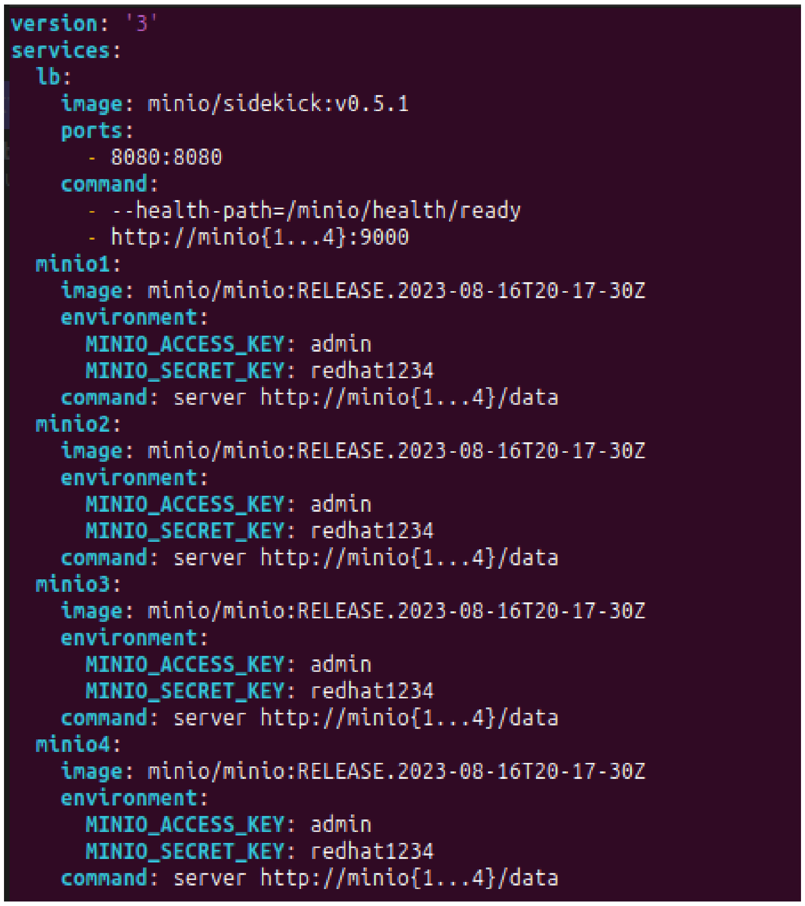
- Press i to enter insert mode, where you can edit the file.
- Use the arrow keys to navigate to the desired location in the file.
- Press Esc to exit insert mode and return to command mode.
- To save your changes, press : to enter command mode, then type w to write (save) the file and q to quit Vim. You can combine these commands as :wq to save and exit.



```
sam@sam-Standard-PC-Q35-ICH9-2009:~$ vim docker-compose.yml
```

4. Add the Following Script to docker-compose.yml:

```
version: '3'
services:
  lb:
    image: minio/sidekick:v0.5.1
    ports:
      - 8080:8080
    command:
      - --health-path=/minio/health/ready
      - http://minio{1...4}:9000
  minio1:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio2:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio3:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio4:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
```



```
version: '3'
services:
  lb:
    image: minio/sidekick:v0.5.1
    ports:
      - 8080:8080
    command:
      - --health-path=/minio/health/ready
      - http://minio{1...4}:9000
  minio1:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio2:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio3:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
  minio4:
    image: minio/minio:RELEASE.2023-08-16T20-17-30Z
    environment:
      MINIO_ACCESS_KEY: admin
      MINIO_SECRET_KEY: redhat1234
    command: server http://minio{1...4}/data
```

5. Start Docker Containers in Detached Mode

```
sudo docker-compose up -d
```

The command **docker-compose up -d** is used to start Docker containers defined in a Docker Compose file in detached mode. Let's discuss the commands in detail:

- **`docker-compose`** : This is the command-line tool for Docker Compose, which is a tool for defining and running multi-container Docker applications.
- **`up`** : This subcommand is used to start the containers defined in the Docker Compose file.
- **`-d` or `--detach`** : This flag tells Docker Compose to run the containers in the background, detached from the terminal. This means that we can continue to use the terminal for other tasks without the container logs and output being displayed in our terminal window. The -d flag is optional, but it's commonly used for long-running services.

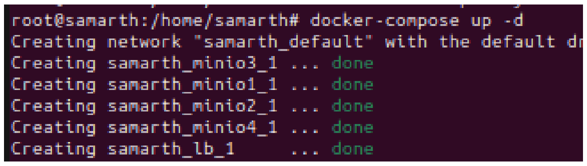
When we run docker-compose up -d, Docker Compose will read the docker-compose.yml file in the current directory (or you can specify a different Compose file with the -f flag), create and start the containers defined in the Compose file, and detach them from the terminal. This is useful for running services or applications that should continue to run in the background.

Here's a basic example of how anyone might use this command:

1. Navigate to the directory containing your docker-compose.yml file.

2. Run docker-compose up -d.

Output:



```
root@samarth:/home/samarth# docker-compose up -d
Creating network "samarth_default" with the default dr
Creating samarth_minio3_1 ... done
Creating samarth_minio1_1 ... done
Creating samarth_minio2_1 ... done
Creating samarth_minio4_1 ... done
Creating samarth_lb_1     ... done
```

6. Edit the /etc/hosts File

```
vim /etc/hosts
```

The command vim /etc/hosts is used to open the /etc/hosts file in the Vim text editor. This file is typically found on Unix-like operating systems, including Linux. The /etc/hosts file is used to map hostnames to IP addresses and is often used for local DNS resolution.

Here's a breakdown of the command:

`vim` : This is the command to start the Vim text editor.

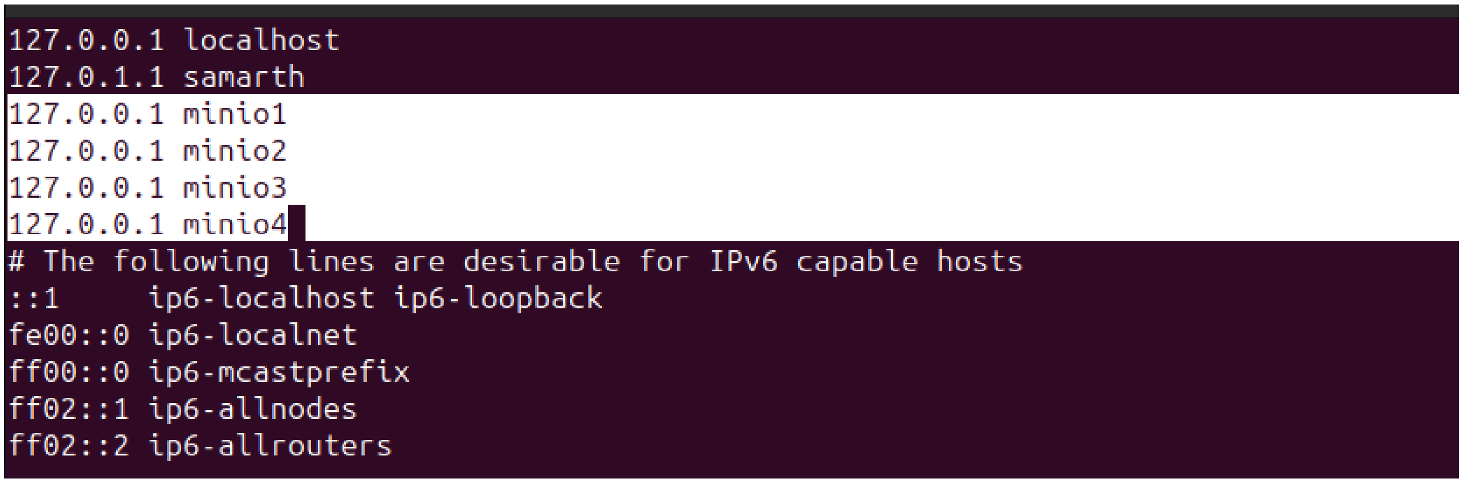
`/etc/hosts` : This is the file path. It specifies the location of the file you want to open. In this case, you are opening the /etc/hosts file, which is commonly used to define static IP address-to-hostname mappings and hostname resolutions.

When you run this command, Vim will open the /etc/hosts file, allowing you to view and edit its contents. Vim is a powerful and customizable text editor, so you can use it to make changes to the file and save those changes if needed.

Input

```
127.0.0.1 minio1
127.0.0.1 minio2
127.0.0.1 minio3
127.0.0.1 minio4
```

Output:



```
127.0.0.1 localhost
127.0.1.1 samarth
127.0.0.1 minio1
127.0.0.1 minio2
127.0.0.1 minio3
127.0.0.1 minio4
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

7. Install the MinIO Client

The MinIO Client allows you to work with your MinIO server from the command line.

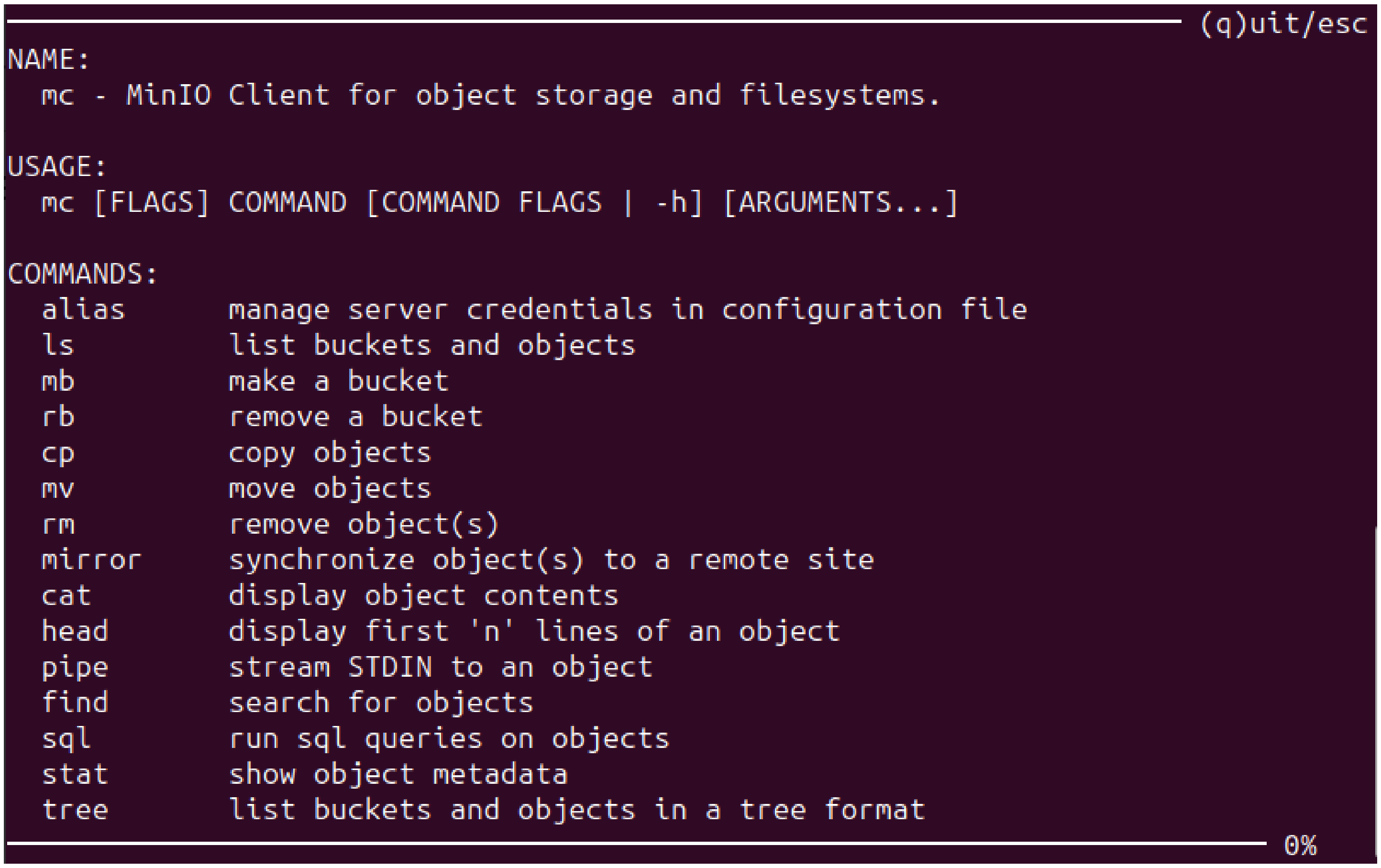
Install the mc command line tool onto the host machine. Click the tab that corresponds to the host machine operating system or environment:

The following commands add a temporary extension to your system PATH for running the mc utility. Defer to your operating system instructions for making permanent modifications to your system PATH.

Alternatively, execute mc by navigating to the parent folder and running ./mc --help

```
curl https://dl.min.io/client/mc/release/linux-amd64/mc \
  --create-dirs \
  -o $HOME/minio-binaries/mc


chmod +x $HOME/minio-binaries/mc


export PATH=$PATH:$HOME/minio-binaries/


  mc --help
```

Output:



```
                                                                   (q)uit/esc
NAME:
  mc - MinIO Client for object storage and filesystems.

USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:
  alias      manage server credentials in configuration file
  ls         list buckets and objects
  mb         make a bucket
  rb         remove a bucket
  cp         copy objects
  mv         move objects
  rm         remove object(s)
  mirror     synchronize object(s) to a remote site
  cat        display object contents
  head       display first 'n' lines of an object
  pipe       stream STDIN to an object
  find       search for objects
  sql        run sql queries on objects
  stat       show object metadata
  tree       list buckets and objects in a tree format
                                                                          0%
```

8. Configure a MinIO Alias

```
mc alias set myminio http://127.0.0.1:8080 admin redhat1234
```

The command you've provided appears to be related to configuring an alias for MinIO, a popular object storage server. This alias is being set using the mc command-line tool. Let me break down the command for you:

`mc` : This is the command-line tool used for interacting with MinIO servers.

`alias set` : This part of the command is used to set up an alias for a MinIO server, allowing you to refer to it by a more convenient name.

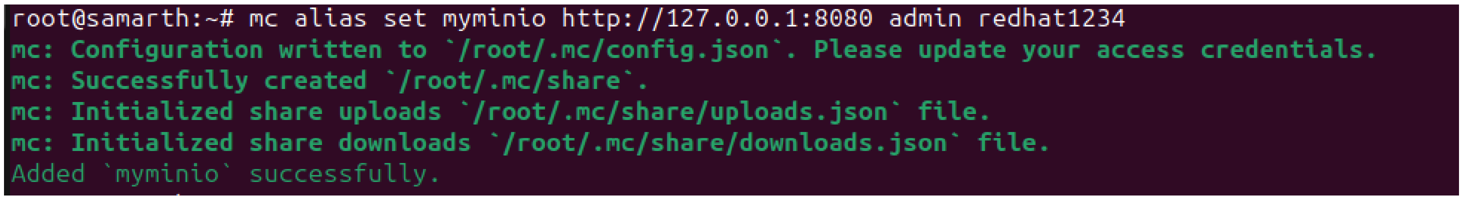
`myminio` : This is the alias you are setting for the MinIO server. You can choose any name you like as long as it's unique and helps you identify the server.

`http://127.0.0.1:8080` : This is the URL of the MinIO server you are configuring. In this case, it's set to a local address, which means you are configuring an alias for a MinIO server running on your local machine at port 8080. You would replace this with the actual URL of your MinIO server if it's hosted elsewhere.

`admin` : This is the access key or username used to authenticate with the MinIO server.

`redhat1234` : This is the corresponding secret key or password used to authenticate with the MinIO server. Make sure to keep this key secure.

Output:



```
root@samarth:~# mc alias set myminio http://127.0.0.1:8080 admin redhat1234
mc: Configuration written to `/root/.mc/config.json`. Please update your access credentials.
mc: Successfully created `/root/.mc/share`.
mc: Initialized share uploads `/root/.mc/share/uploads.json` file.
mc: Initialized share downloads `/root/.mc/share/downloads.json` file.
Added `myminio` successfully.
```

7. Get Information About the MinIO Server

```
mc admin info myminio
```

The command `mc admin info myminio` is used with the `mc` (MinIO Client) tool to retrieve information about a MinIO server instance named "myminio." MinIO is an open-source, high-performance object storage system that can be used for storing and managing large amounts of data.

Here's a breakdown of the command:

`mc` : This is the MinIO Client command-line tool used for interacting with MinIO servers.
admin: This is a subcommand of mc used for performing administrative tasks on a MinIO server.

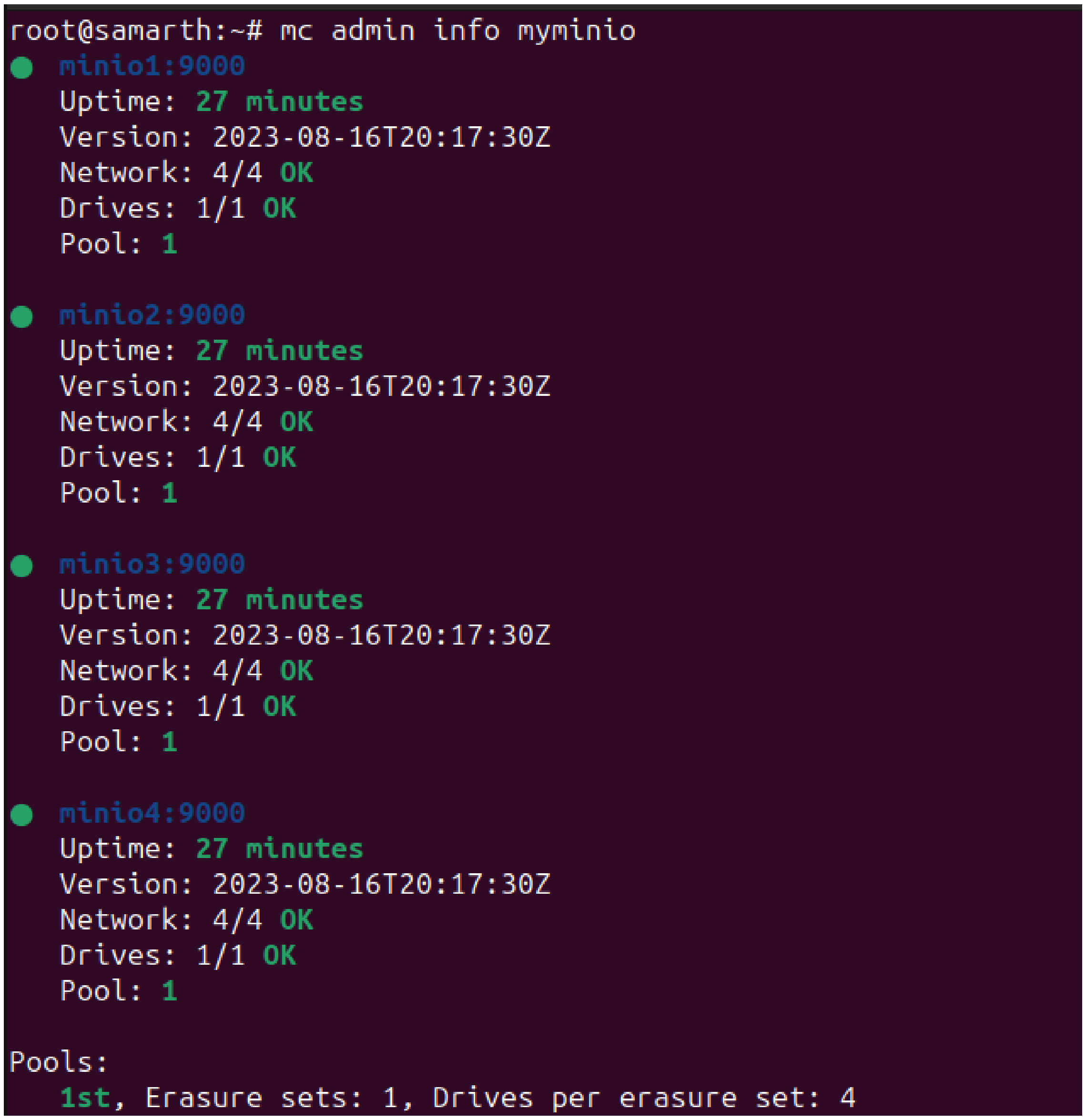
`info` : This is a specific action performed by the admin subcommand. It is used to retrieve various information and statistics about the MinIO server.

`myminio` : This is the alias or configuration name for the MinIO server you want to get information about. The name "myminio" should correspond to a server configuration you've previously set
up using the mc config host add command.

When you run `mc admin info myminio` , the MinIO Client will connect to the MinIO server specified by the "myminio" alias and retrieve information such as server version, configuration settings, and statistics about buckets and objects. The exact information displayed may vary depending on the version of MinIO and the server's configuration.

Please make sure that you have the MinIO Client (`mc`) properly configured with the necessary access credentials and server information before running this command.

Output:



```
root@samarth:~# mc admin info myminio
●  minio1:9000
   Uptime: 27 minutes
   Version: 2023-08-16T20:17:30Z
   Network: 4/4 OK
   Drives: 1/1 OK
   Pool: 1

●  minio2:9000
   Uptime: 27 minutes
   Version: 2023-08-16T20:17:30Z
   Network: 4/4 OK
   Drives: 1/1 OK
   Pool: 1

●  minio3:9000
   Uptime: 27 minutes
   Version: 2023-08-16T20:17:30Z
   Network: 4/4 OK
   Drives: 1/1 OK
   Pool: 1

●  minio4:9000
   Uptime: 27 minutes
   Version: 2023-08-16T20:17:30Z
   Network: 4/4 OK
   Drives: 1/1 OK
   Pool: 1

Pools:
   1st, Erasure sets: 1, Drives per erasure set: 4
```

9. Execute a Command Inside a Docker Container

```
sudo docker exec -it root_lb_1 "/sidekick" -a :8989 --health-path=/minio/health/ready http://minio{
```

The above command is a Docker command that uses the 'docker exec' command to run a command inside a Docker container. Let's break down the command:

1. `docker exec` : This is the Docker command used to execute a command inside a running Docker container.
2. `-it` : These are options for the docker exec command:

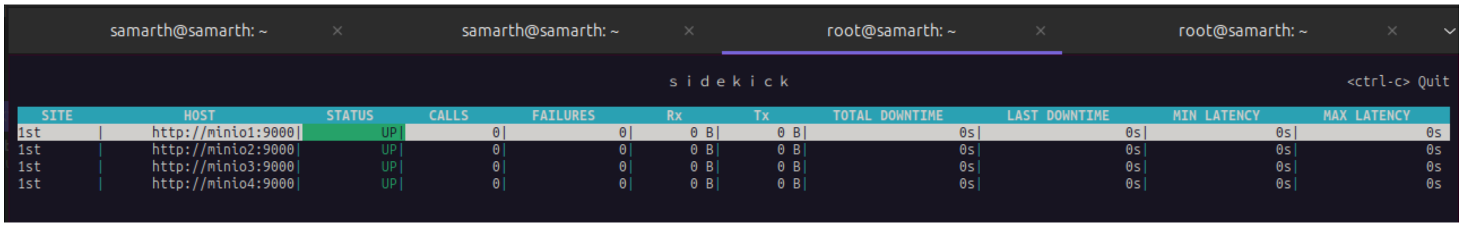
- -i: Keep STDIN open even if not attached.
- -t: Allocate a pseudo-TTY.

3. These options allow you to interact with the command being executed inside the container.
4. `root_lb_1` : This is the name or ID of the Docker container in which you want to execute the command. In this case, the container is named "root_lb_1."

5. **`"/sidekick"`** : This is the command that you want to run inside the specified container. It appears to be running the "/sidekick" executable or script inside the container.
6. **`-a :8989`** : This part of the command specifies an option for the "/sidekick" command being executed inside the container. It sets the address to ":8989."
7. **`--health-path=/minio/health/ready`** : Another option for the "/sidekick" command, it specifies the health check path as "/minio/health/ready."
8. **`http://minio{1...4}:9000`** : This is part of the command passed as an argument to "/sidekick." It appears to be a list of URLs, generated using brace expansion (from minio1 to minio4), which point to different Minio servers on port 9000.

In summary, this Docker command runs the "/sidekick" command inside the "root_lb_1" container, configuring it to use the address ":8989" and specifying a health check path. The "/sidekick" command appears to be designed to interact with multiple Minio servers (minio1 to minio4) on port 9000, possibly for some kind of load balancing or health checking operation.

Output:



```
                                      s i d e k i c k                                                  <ctrl-c> Quit
SITE         HOST           STATUS     CALLS       FAILURES       Rx        Tx     TOTAL DOWNTIME   LAST DOWNTIME   MIN LATENCY   MAX LATENCY
1st    |   http://minio1:9000|    UP|          0|           0|       0 B|      0 B|              0s|             0s|           0s|           0s
1st    |   http://minio2:9000|    UP|          0|           0|       0 B|      0 B|              0s|             0s|           0s|           0s
1st    |   http://minio3:9000|    UP|          0|           0|       0 B|      0 B|              0s|             0s|           0s|           0s
1st    |   http://minio4:9000|    UP|          0|           0|       0 B|      0 B|              0s|             0s|           0s|           0s
```

10. View Docker Container Logs in Real-Time

```
sudo docker logs -f samarth_minio2_1
```

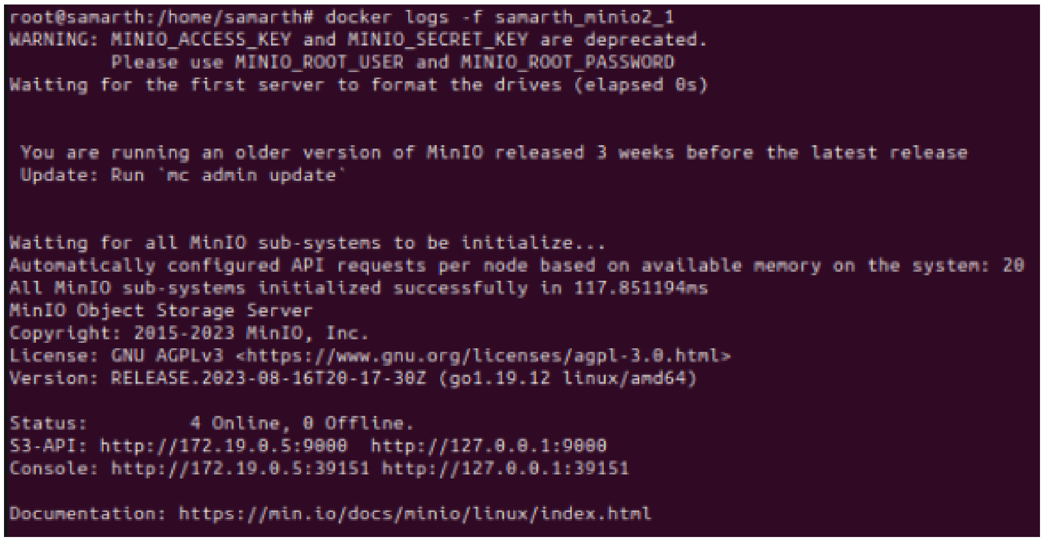
The `docker logs -f` command is used to view the logs of a Docker container in real-time. Here's a breakdown of the command you provided:

- **`docker`** : This is the command-line tool for managing Docker containers.
- **`logs`** : This is the subcommand that tells Docker you want to view the logs of a container.
- **`-f or --follow`** : This flag tells Docker to follow the log output in real-time, similar to the tail -f command in Unix-like systems.

After the -f flag, you have the name of the Docker container you want to view logs for, in this case, samarth_minio2_1. This is typically the name of a running Docker container.

So, when you run docker logs -f samarth_minio2_1, you'll see the logs from the samarth_minio2_1 container as they are generated in real-time. This can be useful for debugging, monitoring, or troubleshooting Docker containers. To exit from viewing the logs, you can usually press Ctrl+C.

Output:

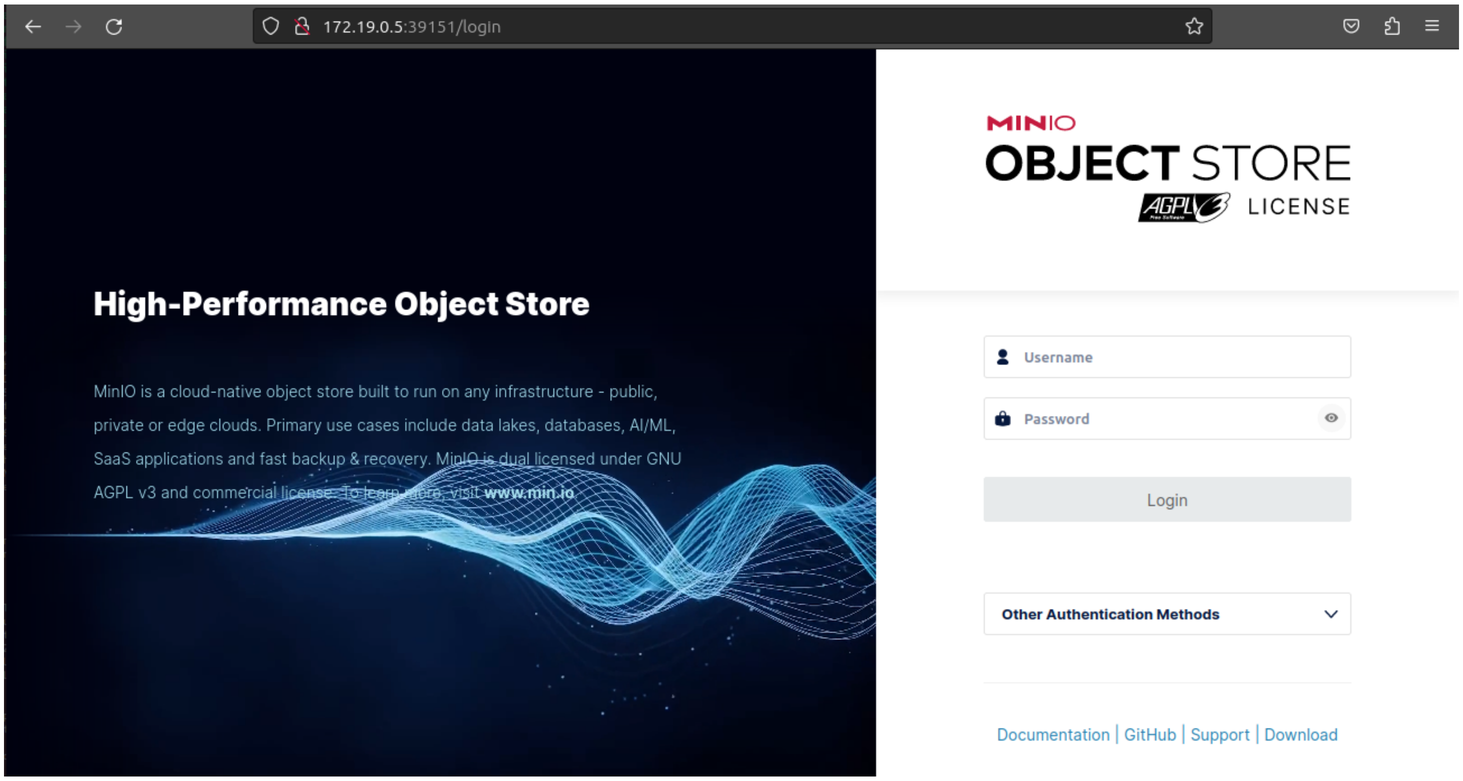


```
root@samarth:/home/samarth# docker logs -f samarth_minio2_1
WARNING: MINIO_ACCESS_KEY and MINIO_SECRET_KEY are deprecated.
         Please use MINIO_ROOT_USER and MINIO_ROOT_PASSWORD
Waiting for the first server to format the drives (elapsed 0s)


 You are running an older version of MinIO released 3 weeks before the latest release
 Update: Run `mc admin update`


Waiting for all MinIO sub-systems to be initialize...
Automatically configured API requests per node based on available memory on the system: 20
All MinIO sub-systems initialized successfully in 117.851194ms
MinIO Object Storage Server
Copyright: 2015-2023 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2023-08-16T20-17-30Z (go1.19.12 linux/amd64)

Status:         4 Online, 0 Offline.
S3-API: http://172.19.0.5:9000  http://127.0.0.1:9000
Console: http://172.19.0.5:39151 http://127.0.0.1:39151

Documentation: https://min.io/docs/minio/linux/index.html
```

11. Open a Web Page

Open the following link in your browser: http://172.19.0.5:39151

Output:

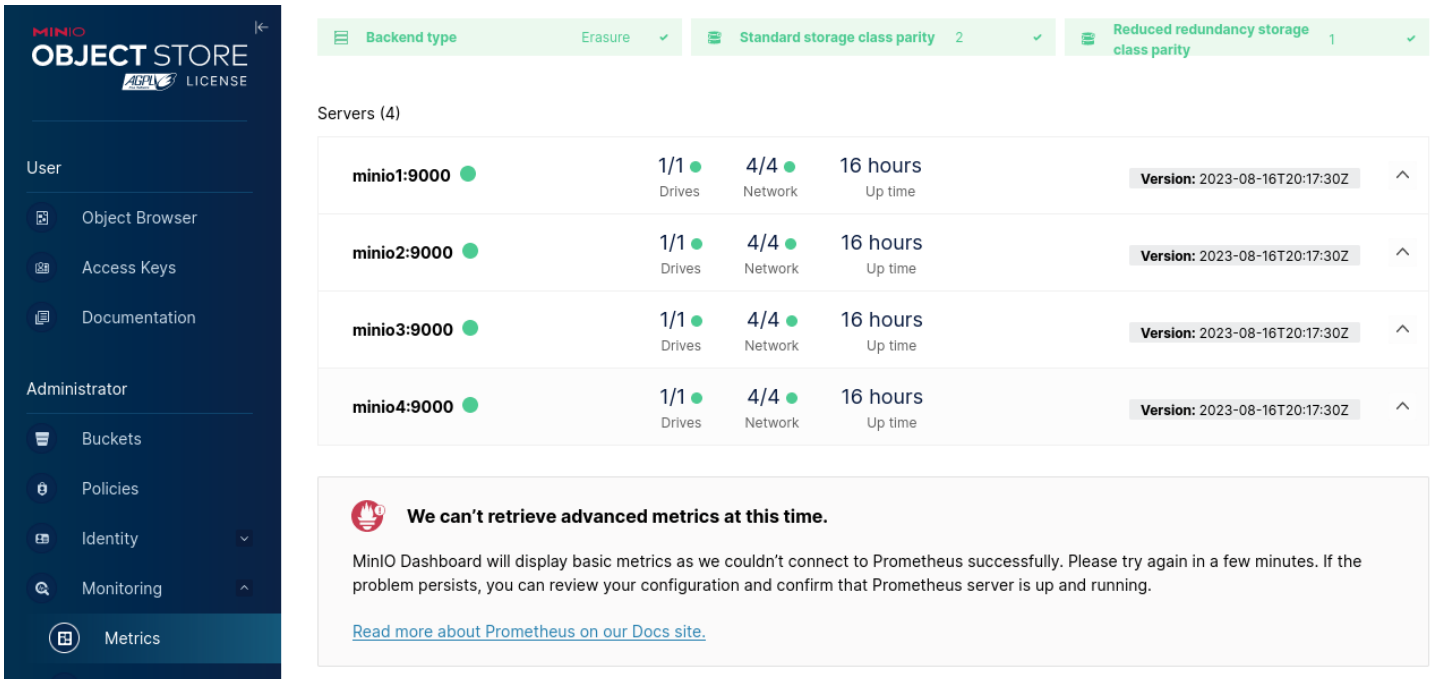


High-Performance Object Store

MinIO is a cloud-native object store built to run on any infrastructure - public, private or edge clouds. Primary use cases include data lakes, databases, AI/ML, SaaS applications and fast backup & recovery. MinIO is dual licensed under GNU AGPL v3 and commercial license. To learn more, visit www.min.io

12. Open metrics

All Nodes are shown in the below screenshot.



Backend type Erasure ✓ | Standard storage class parity 2 ✓ | Reduced redundancy storage class parity 1 ✓

Servers (4)

Server	Drives	Network	Up time	Version
minio1:9000 ●	1/1 ●	4/4 ●	16 hours	Version: 2023-08-16T20:17:30Z
minio2:9000 ●	1/1 ●	4/4 ●	16 hours	Version: 2023-08-16T20:17:30Z
minio3:9000 ●	1/1 ●	4/4 ●	16 hours	Version: 2023-08-16T20:17:30Z
minio4:9000 ●	1/1 ●	4/4 ●	16 hours	Version: 2023-08-16T20:17:30Z

We can't retrieve advanced metrics at this time.

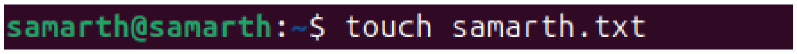
MinIO Dashboard will display basic metrics as we couldn't connect to Prometheus successfully. Please try again in a few minutes. If the problem persists, you can review your configuration and confirm that Prometheus server is up and running.

Read more about Prometheus on our Docs site.

13. Create a New Empty File

Use the touch command to create a new empty file named "samarth.txt."

touch samarth.txt



```
samarth@samarth:~$ touch samarth.txt
```

14. Create a MinIO Bucket

Use the `mc mb` command to create a new MinIO bucket.

Example:

```
mc mb myminio/sam1
```

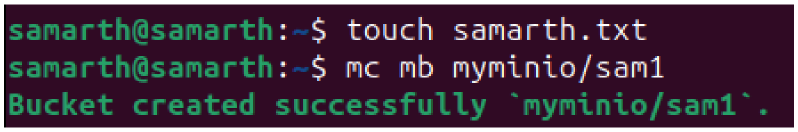
The command `mc mb myminio/sam1` is used with the MinIO Client (mc) tool to create a new bucket named "sam1" in the MinIO server configured with the alias "myminio."

Here's a breakdown of the command:

- **`mc`** : This is the MinIO Client command-line tool.
- **`mb`** : This is a subcommand of mc used for creating a new bucket.
- **`myminio`** : This is the alias or configuration name for the MinIO server you want to interact with.
- **`sam1`** : This is the name of the bucket you want to create.

When you run this command, it will create a new bucket named "sam1" in the MinIO server configured with the alias "myminio." Buckets in MinIO are used to organise and store objects (files or data).

Output:



```
samarth@samarth:~$ touch samarth.txt
samarth@samarth:~$ mc mb myminio/sam1
Bucket created successfully `myminio/sam1`.
```

15.The object to copy from bucket

```
./mc cp /home/samarth/screenshots/sr.png -/myminio/sam1
```

The above command is using mc, which stands for MinIO Client. MinIO is an open-source, object storage server, and mc is a command-line tool for interacting with MinIO servers.

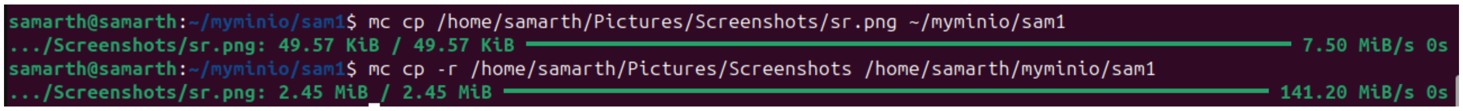
Here's a breakdown of the command:

`mc` : This is the MinIO Client command-line tool.

`cp` : This is the command to copy files or objects.

`/home/samarth/screenshots/sr.png` : This is the source file or object that you want to copy. It's located in the /home/samarth/screenshots/ directory and is named sr.png.

`-/myminio/sam1` : This is the destination where you want to copy the source file or object. It appears to be specifying a MinIO server named myminio and a bucket or directory within that server named sam1.



```
samarth@samarth:~/myminio/sam1$ mc cp /home/samarth/Pictures/Screenshots/sr.png ~/myminio/sam1
.../Screenshots/sr.png: 49.57 KiB / 49.57 KiB ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.50 MiB/s 0s
samarth@samarth:~/myminio/sam1$ mc cp -r /home/samarth/Pictures/Screenshots /home/samarth/myminio/sam1
.../Screenshots/sr.png: 2.45 MiB / 2.45 MiB ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.20 MiB/s 0s
```

16.All Images are shown in the below screenshot

```
mc ls myminio/sam1
```

The command mc ls myminio/sam1 is used with the MinIO Client (mc) to list the objects or files within the sam1 bucket or directory on the MinIO server named myminio.

When you run this command, it will list the objects/files in the specified MinIO bucket, displaying their names, sizes, and other relevant information.

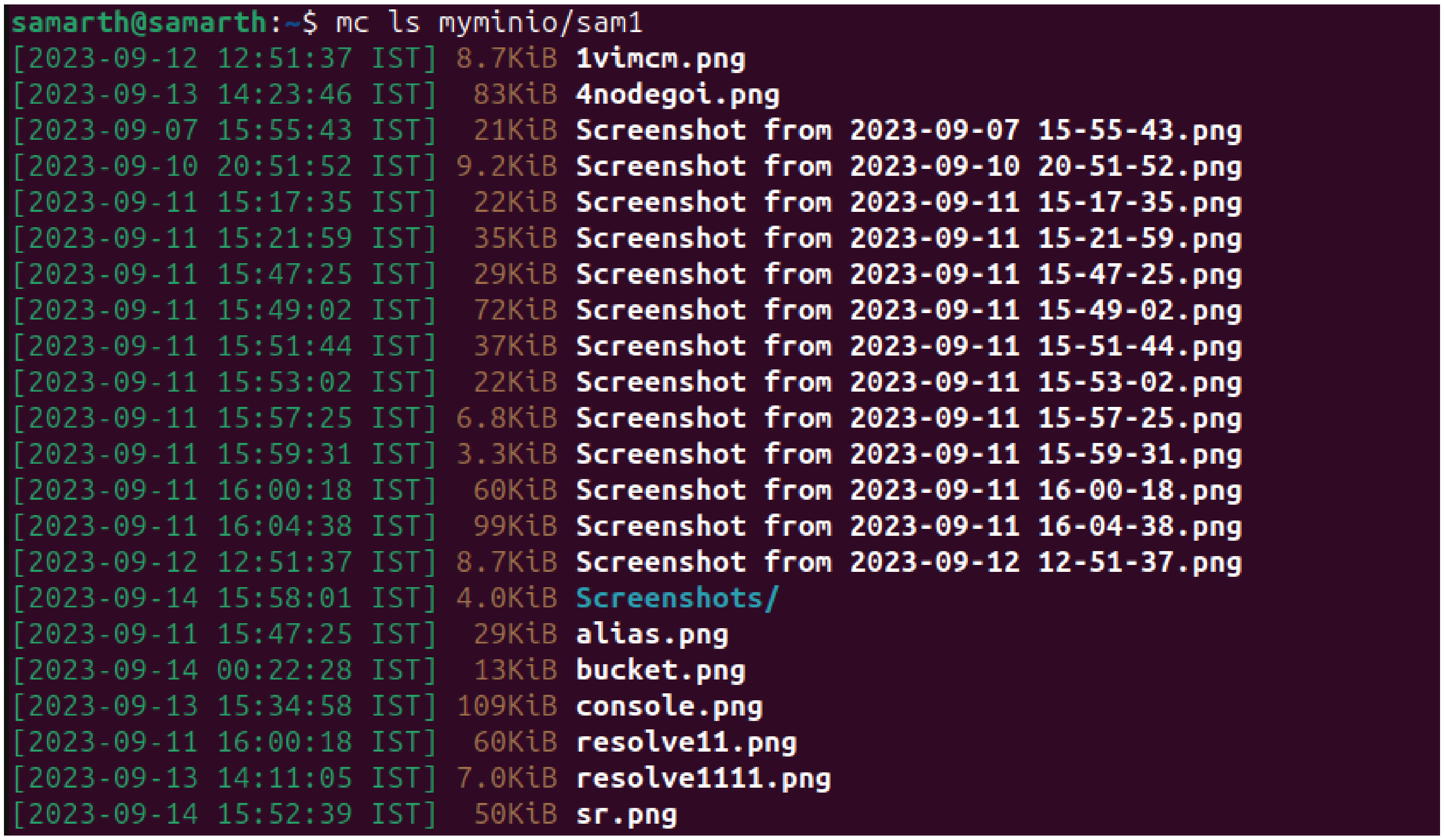
Here's a breakdown of the command:

`mc` : The MinIO Client command-line tool.

`ls` : The subcommand to list objects or files.

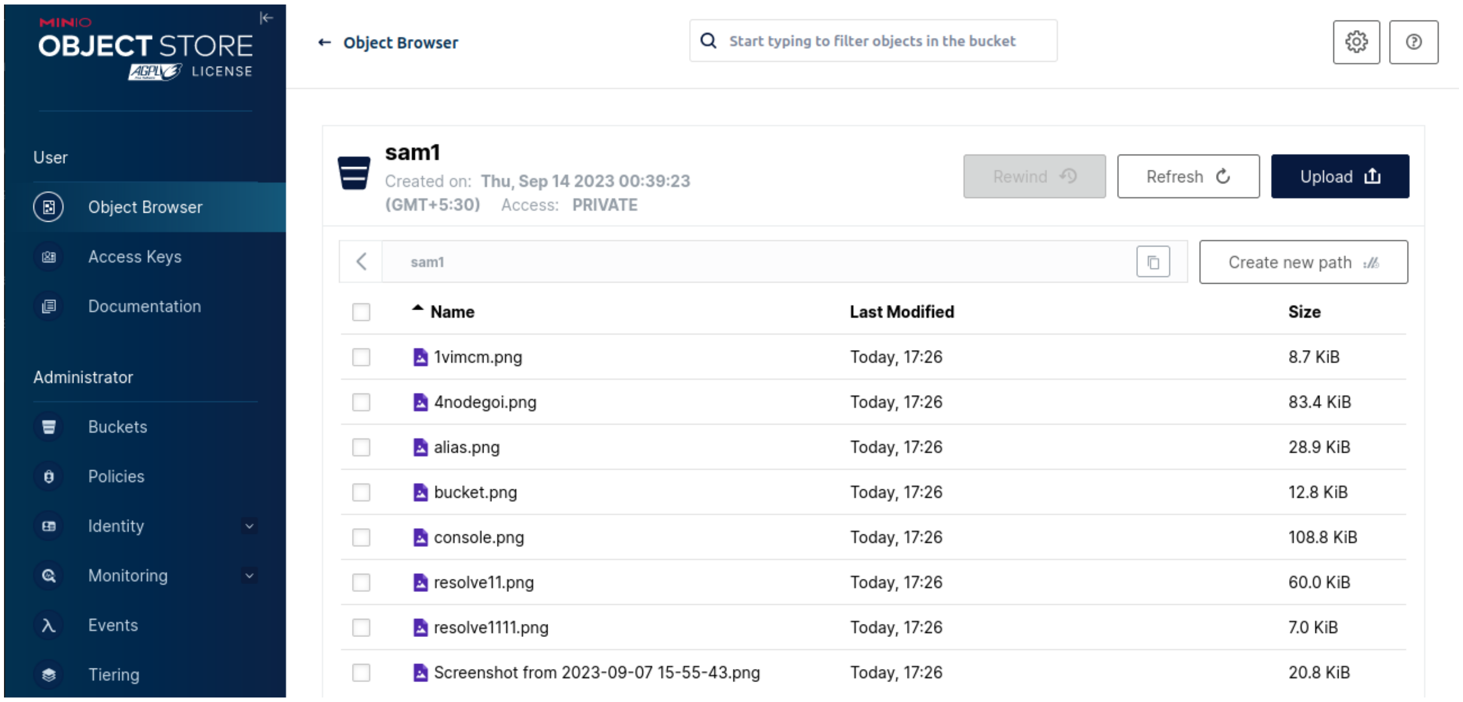
`myminio/sam1:` The path to the bucket or directory on the MinIO server where you want to list objects.

Output:



```
samarth@samarth:~$ mc ls myminio/sam1
[2023-09-12 12:51:37 IST] 8.7KiB 1vimcm.png
[2023-09-13 14:23:46 IST]  83KiB 4nodegoi.png
[2023-09-07 15:55:43 IST]  21KiB Screenshot from 2023-09-07 15-55-43.png
[2023-09-10 20:51:52 IST] 9.2KiB Screenshot from 2023-09-10 20-51-52.png
[2023-09-11 15:17:35 IST]  22KiB Screenshot from 2023-09-11 15-17-35.png
[2023-09-11 15:21:59 IST]  35KiB Screenshot from 2023-09-11 15-21-59.png
[2023-09-11 15:47:25 IST]  29KiB Screenshot from 2023-09-11 15-47-25.png
[2023-09-11 15:49:02 IST]  72KiB Screenshot from 2023-09-11 15-49-02.png
[2023-09-11 15:51:44 IST]  37KiB Screenshot from 2023-09-11 15-51-44.png
[2023-09-11 15:53:02 IST]  22KiB Screenshot from 2023-09-11 15-53-02.png
[2023-09-11 15:57:25 IST] 6.8KiB Screenshot from 2023-09-11 15-57-25.png
[2023-09-11 15:59:31 IST] 3.3KiB Screenshot from 2023-09-11 15-59-31.png
[2023-09-11 16:00:18 IST]  60KiB Screenshot from 2023-09-11 16-00-18.png
[2023-09-11 16:04:38 IST]  99KiB Screenshot from 2023-09-11 16-04-38.png
[2023-09-12 12:51:37 IST] 8.7KiB Screenshot from 2023-09-12 12-51-37.png
[2023-09-14 15:58:01 IST] 4.0KiB Screenshots/
[2023-09-11 15:47:25 IST]  29KiB alias.png
[2023-09-14 00:22:28 IST]  13KiB bucket.png
[2023-09-13 15:34:58 IST] 109KiB console.png
[2023-09-11 16:00:18 IST]  60KiB resolve11.png
[2023-09-13 14:11:05 IST] 7.0KiB resolve1111.png
[2023-09-14 15:52:39 IST]  50KiB sr.png
```

MinIO bucket Output:



sam1
Created on: Thu, Sep 14 2023 00:39:23 (GMT+5:30) Access: PRIVATE

sam1

Name	Last Modified	Size
1vimcm.png	Today, 17:26	8.7 KiB
4nodegoi.png	Today, 17:26	83.4 KiB
alias.png	Today, 17:26	28.9 KiB
bucket.png	Today, 17:26	12.8 KiB
console.png	Today, 17:26	108.8 KiB
resolve11.png	Today, 17:26	60.0 KiB
resolve1111.png	Today, 17:26	7.0 KiB
Screenshot from 2023-09-07 15-55-43.png	Today, 17:26	20.8 KiB

Reference Link

For more details, you can refer to this GitHub link.