**Cloud Computing Virtualization Management System Documentation**

**Menna El Banna 222745**

**Nada Mostafa 221427**

**Mariam Moataz 223675**

**Ahmed Ayman 221341**

**Shahd Ali 224217**

**Shahd Wael 222045**

1. **Introduction**

The Cloud Computing Virtualization Management System (CCVMS) is a comprehensive platform developed to efficiently manage virtualized computing resources. The system uses Flask for its backend API, QEMU , docker desktop for virtual machine (VM) and virtual disk management, and docker files management by Python for the backend logic, providing users with a seamless and intuitive way to interact with virtualized environments. The platform supports a wide array of features, including the creation, configuration, and management of virtual disks and virtual machines. Also create and management docker files , images , and containers

CCVMS enables users to perform tasks such as creating virtual disks, resizing them, deleting or converting between disk formats, as well as creating and managing virtual machines. Also docker files , images , and containers ,

The system is built to support scalability, resource allocation, and dynamic interactions with virtual environments, making it an ideal tool for those looking to manage virtualized resources effectively.

1. **Core Features**

**2.1 Virtual Disk Management**

The following features are included:

* **Create Virtual Disk:** Users can create virtual disks by specifying size and format (e.g., QCOW2, RAW).
* **Resize Virtual Disk:** The system allows resizing of virtual disks, enabling users to modify storage size as needed.
* **Delete Virtual Disk**: When a virtual disk is no longer needed, users can delete it to free up resources.
* **Get Disk Information:** Users can retrieve detailed information about each disk, such as size, format, and other properties, through the GET /disk\_info API endpoint.
* **Disk Conversion:** The platform supports converting virtual disks between different formats (e.g., QCOW2 to RAW) for compatibility with various virtualization platforms. This can be done using the POST /convert\_disk API endpoint.

**2.2 Virtual Machine (VM) Management**

VM management is another key feature, offering the following capabilities:

* **Create Virtual Machine:** Users can create VMs using existing virtual disks or ISO images. They can configure resources such as CPU, RAM, and disk space according to their needs. The system can be integrated with the POST /create\_vm API endpoint for VM creation.
* **Boot VM from ISO:** Users can boot a VM directly from an ISO image, making it easy to install an operating system or other software within the VM.
* **Modify VM Resources:** The system allows for dynamic resource adjustments (CPU, RAM) to ensure optimal performance based on workload demands. Users can modify these resources using the backend API to manage virtual resources.
* **Start and Stop VMs:** Users can start and stop virtual machines as needed, ensuring resources are used efficiently. This functionality is available via the appropriate API endpoints for starting and stopping VMs.
* **Delete VM:** Unused or unnecessary VMs can be deleted through the POST /delete\_vm API endpoint, helping maintain a clean system environment and free up resources.

**2.3 VM Resource Allocation**

Proper resource allocation is essential for VM performance:

* **CPU Allocation:** Users can allocate virtual CPUs to VMs, ensuring sufficient processing power for their tasks. This allows efficient resource management and ensures that VMs perform optimally.
* **RAM Allocation:** The system allows users to allocate specific amounts of RAM to each VM. This flexibility enables efficient memory management, ensuring that each VM has the necessary resources to run applications without impacting overall system performance.

2.4 docker

* **Create Docker File**: user enables to create docker file for their docker images.
* **Build Docker Image** : user can create docker image. In the docker files he has and image
* **List Docker Images**: user can list all the docker images on the system.
* **List all the running containers** : users can list all the running containers.
* **Stop a container**: users can stop specific container
* **Remove container** : users can remove a specific container
* Remove while stop container : users can remove a specific container when they stop it
* **Search Image** : system allows the users to search for a docker image by name locally from his images or from docker hub .
* **Download/Pull image**: system allow the users to download a docker image from DockerHub

**2.4 API Interface for System Interaction**

The backend of CCVMS is built with Flask, and it exposes a RESTful API that enables users to interact with the system programmatically. Key API endpoints include:

|  |  |  |
| --- | --- | --- |
| Features in: | Post | Get |
| disk | * **POST /create\_disk:** Creates a new virtual disk with specified size and format. * **POST /resize\_disk:** Resizes an existing virtual disk, allowing users to modify its storage capacity. * **POST /delete\_disk:** Deletes a specified disk. * **POST /convert\_disk:** Converts a disk between different formats (e.g., QCOW2 to RAW). | * **GET /disk\_info:** Retrieves detailed information about a specific disk, such as its size, format, and properties. * **GET /list\_disks:** Lists all available disks in the system, allowing users to see the available storage options for creating VMs. |
| vm | * **POST /save \_vm\_config:** save virtual machine with a specified disk and resource configuration. * **POST /edit \_vm\_config:** Edits an existing virtual machine’s settings. * **POST /run\_vm:** run a virtual machine * **POST /delete\_vm:** Deletes a specified virtual machine. | * **GET /list\_vms:** Lists all available vms in the system, allowing users to see the available storage options for creating VMs. |
| Docker | * **/create-dockerfile** :route accepts a folder name and Dockerfile content, then saves as a Dockerfile under the specified directory * **/build-image** : route builds a Docker image from a Dockerfile located in a specified folder * **/stop-container and /start-container** :routes are used to stop and start Docker containers respectively, using the container ID * **/create-container** : route allows users to create and run a new Docker container from a specified image * **/search-local-image**: route searches locally available Docker images based on a query string, filtering results by name or tag. **/search-dockerhub-image**: route queries Docker Hub for repositories that match a given search term, returning details * **/pull-docker-image**: route pulls an image from Docker Hub using the provided image name. **/remove-container** :route removes a stopped container, and the **/stop-remove-container** route first stops a running container and then removes it | * **/list-images route**: returns a list of all locally available Docker images along with metadata  /**list-running-containers**: route lists only currently running Docker containers, showing details * **/list-all-containers**: route lists all containers on the system, whether running or stopped, along with their status and associated image details. |

1. **Technologies Used**

**3.1 Flask**

Flask is a lightweight Python web framework used to build the backend of the CCVMS. It allows for rapid development of RESTful APIs, making it a perfect choice for this system.

**3.2 QEMU**

QEMU (Quick Emulator) is an open-source virtualization tool that emulates various hardware configurations. It is used in CCVMS to create and manage virtual machines and virtual disks.

**3.3 docker desktop**

a GUI application developed by Docker Inc. that simplifies the use of Docker on Windows, macOS, and Linux (beta). It provides an easy-to-use interface for managing containers, images, volumes, networks, and more.

**3.4 Python**

Python is the main programming language for the system’s backend logic. It interacts with QEMU for VM and disk management, and it handles the API logic with Flask.

**3.5 JSON**

JSON (JavaScript Object Notation) is used for data exchange between the backend and clients. The Flask API sends and receives data in JSON format, which is lightweight and easy to process, making it an ideal choice for this system.

1. **Summary**

The Cloud Computing Project is designed to provide users with the ability to manage and create virtual resources, including virtual disks, virtual machines (VMs),and docker files, images, and containers within a cloud environment. The project focuses on providing flexibility and control over virtualized resources, catering to users who require customization and scalability for their computing needs. The system enables users to create virtual disks, configure virtual machines, and specify parameters such as CPU, memory, and disk allocation, also creating and managing docker files, images, and containers specifying some parameters offering a powerful tool for managing cloud resources.

Key features of the project include:

* **Virtual Disk Creation:** The platform allows users to create virtual disks by specifying parameters like type, size, and format. This flexibility enables users to define storage solutions according to their specific needs.
* **Virtual Machine Configuration: Users** can create virtual machines by specifying configuration details, such as CPU, memory, and disk size. The system supports interactive user input to ensure ease of use and customization.
* **Integration with Virtual Disks:** The virtual machine creation process is seamlessly integrated with the virtual disk management feature, enabling users to select any previously created virtual disks to allocate storage for their virtual machines.
* **Docker file , image , container creation**

The platform allows users to create Docker file , image , containerby specifying parameters like name , docker files content . This flexibility enables users to manage system as list,pull , search , delete , stop , start.

1. **Results**

The implementation of the Cloud Computing Project has successfully met its core objectives, providing users with a robust and flexible platform for managing virtual resources. The key results and outcomes are as follows:

**5.1 Virtual Disk Creation:**

Users can now create virtual disks with a variety of configurations, including selection of type (e.g., SSD, HDD), size, and format. The feature allows for efficient allocation of storage resources, providing flexibility in setting up disk storage according to different requirements.

The system validates user inputs, ensuring the correctness of specified parameters (e.g., ensuring that disk size is within permissible limits).

**5.2 Virtual Machine Configuration:**

Users are able to successfully create virtual machines by specifying detailed configuration parameters such as CPU, memory, and disk size. The system has been optimized to handle varying resource allocation scenarios, ensuring the virtual machine meets the user's requirements.

The interactive input interface simplifies the process for users, allowing them to select and modify resources as per their needs. The feature ensures that users can easily create VMs without requiring advanced technical knowledge.

**5.3 Integration with Virtual Disks:**

The integration between virtual disk creation and virtual machine configuration is seamless. Users can select any of the created virtual disks for use in their virtual machines, ensuring efficient storage allocation and simplified resource management.

This integration ensures that all created virtual resources (disks and VMs) are dynamically linked, making the process of managing virtual infrastructure intuitive and efficient.

**5.4 docker management :**

The Docker API provides full control over Docker operations via HTTP requests, including creating Dockerfiles, building images, listing local and remote images, running, stopping, starting, and removing containers, local image search, Docker Hub image search, and pulling images from Docker Hub. This user-friendly interface can be integrated into a web-based Docker GUI for developers or system administrators.

**5.5 User Experience:**

The system has been well-received by users, with positive feedback on the ease of use and the flexibility provided in configuring virtual resources. The interactive features have been praised for simplifying the process of managing cloud infrastructure, making it accessible even to users with limited technical expertise.