SOFTWARE REQUIREMENTS SPECIFICATION
VERSION 1.0

CLOUDFORGE

NKANSAH NOEL GERHARD YAW
ABOTSI BENJAMIN ETORNAM
OSEI-ABORAH WILLIAM
SOBBIN FLAVIO NANA BADU
LARBI NANA KWASI OPARE
SEREBOUR JASON
QUAYE MARK TETTEH
ADDAE SAMUEL NYARKO

JUNE 20TH, 2025

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS OF
COE 356 SOFTWARE ENGINEERING

REVISION HISTORY

Date	Version	Description	Author
<20/06/2025>	<1.0>	SRS 1.0	Group-5

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

The Purpose of CloudForge is to create a local cloud platform as a service that makes application deployment especially from Africa more efficient. CloudForge aims at allowing developers to deploy their apps with ease as well as enabling users to create and manage Virtual Machines.

1.2 Document Convections

This document contains the following conventions.

Term	Meaning
PVE	Proxmox Virtual Environment
VM	Virtual Machine
VE	Virtual Environment
API	Application Programming Interface
CLI	Command Line Interface
SSH	Secure Shell

1.3 Intended Audience and Reading Suggestions

This document targets the following class of individuals:
Developers using CloudForge APIs and deployment tools
DevOps engineers managing resources
Business stakeholders tracking billing and user activity
End-users managing their virtual machines

1.4 Project Scope

The purpose of this project is to provide a cloud management platform that provides users with VMs or Storage with an easy-to-use Interface. It would include a Proxmox VM cluster or alternatives also includes a database to store user information.

1.5 References

Proxmox VE API Documentation (v8.x) - https://pve.proxmox.com/pve-docs/
IEEE Std 830-1998 - Recommended Practice for Software Requirements Specifications - http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

2 OVERALL DESCRIPTION

2.1 Product Perspective

CloudForge is built on a centralized control plane with multiple Proxmox clusters working as backend nodes. It provides a web-based UI and APIs for interaction.

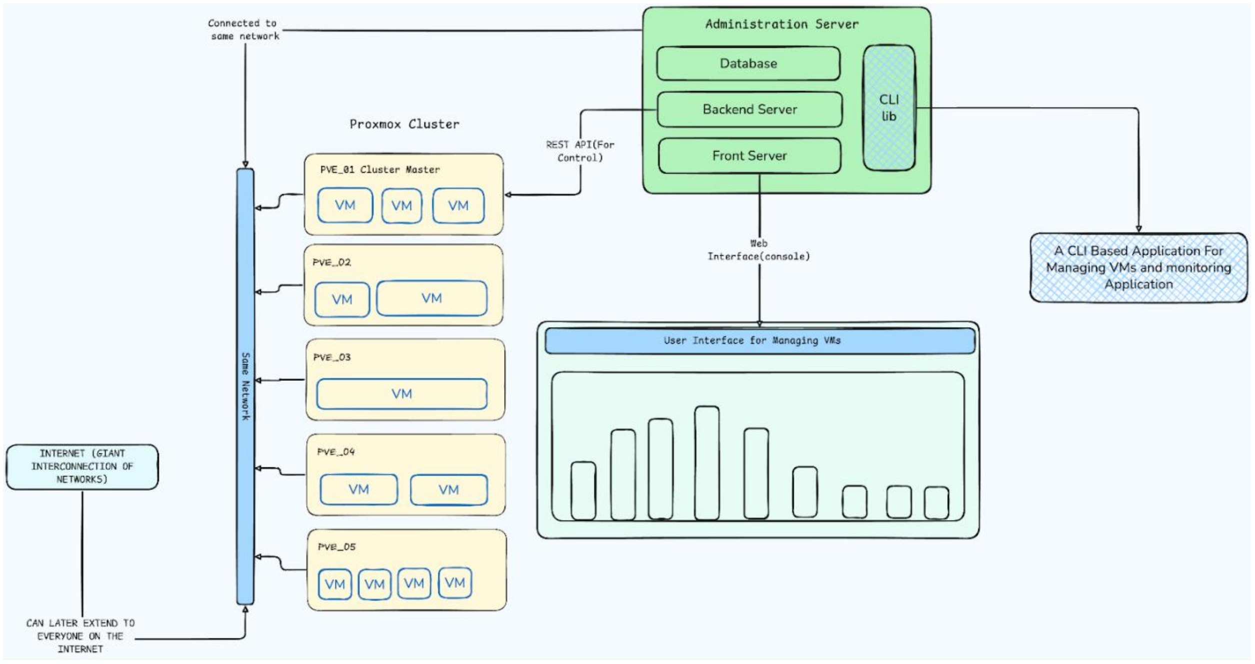

Connected to same network
Administration Server
Database
Backend Server
Front Server
CLI lib
Proxmox Cluster
REST API(For Control)
PVE_01 Cluster Master
VM
VM
VM
PVE_02
VM
VM
PVE_03
VM
PVE_04
VM
VM
PVE_05
VM
VM
VM
VM
Same Network
Web Interface(console)
User Interface for Managing VMs
A CLI Based Application For Managing VMs and monitoring Application
INTERNET (GIANT INTERCONNECTION OF NETWORKS)
CAN LATER EXTEND TO EVERYONE ON THE INTERNET

Figure 1: System Environment Overview

2.2 Product Features

2.2.1 User Account Management

- User Registration: An unauthenticated user shall be able to register for a new account using an email address and password. The system shall send a verification email.
- User Login: A registered user shall be able to log in to the platform using their credentials to obtain an authenticated session. (Website or CLI)
- Password Reset: A user shall be able to request a password reset via their registered email.
- User Profile: An authenticated user shall be able to view and update their profile information (name, contact details, etc.).

2.2.2 Administrator's Dashboard & Control Panel

- System Overview: The administrator shall have a dashboard displaying key metrics of the entire cluster: total CPU/RAM/Storage usage, number of active VMs, and number of registered users.
- Node Management: The administrator shall be able to view the status of each Proxmox node in the cluster, including its individual resource utilization.
- User Management: The administrator shall be able to view, search, suspend, and delete any user account.
- Global VM Management: The administrator shall be able to view and manage (start, stop, delete) any VM from any user on the platform. In case of the emergency.

2.2.3 Customer VM Lifecycle Management

- Create VM: An authenticated user shall be able to provision a new VM. The creation process involves selecting an OS template, a resource plan (e.g., Small, Medium, Large), and providing a hostname. The system shall automatically select the best Proxmox node for placement.

- List VMs: A user shall be able to view a list of all their active and inactive VMs, showing their name, IP address, status (Running/Stopped), and the node it is on.
- VM Actions: For each of their VMs, a user shall be able to perform the following actions: Start, Shutdown, Reboot, and Delete.
- Access Console: A user shall be able to access the noVNC or SPICE console for their VMs directly through the web portal.
- View VM Details: A user shall be able to view detailed information for a specific VM, including its assigned IP address, CPU usage, RAM usage, and disk usage.

2.3 User Classes and Characteristics

Users of the system should be able to set up Virtual Environments which they can use hassle-free. That feature is available to end-users. Administrators for this product will be able to monitor clusters of virtual machines and manage the performance and resources allocated to each Virtual Machine.

2.4 Operating Environment

The system will operate on dedicated servers housed in a datacentre with major African internet hubs.

2.5 Design and Implementation Constraints

Proxmox requires subscriptions from developers to maintain and grow its codebase and functionalities. Hence, this project is built under limitations (less features) from Proxmox.

2.6 Assumptions and Dependencies

The system is assumed to be built and deployed on a dedicated server machine.

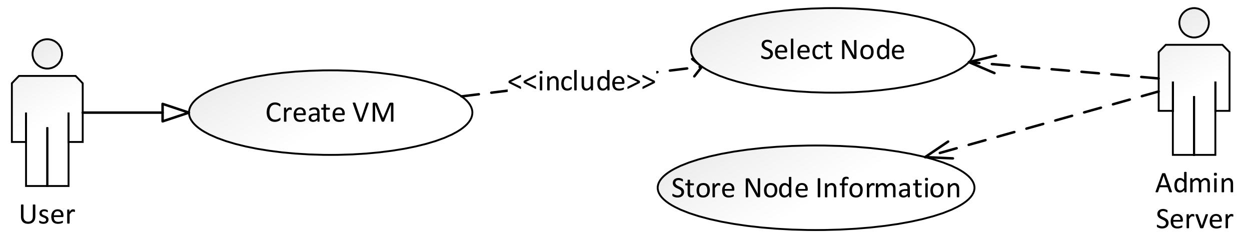
3 SYSTEM FEATURES

3.1 Functional Requirements

3.1.1 User Use Cases:

Use case: Create VM

Diagram:

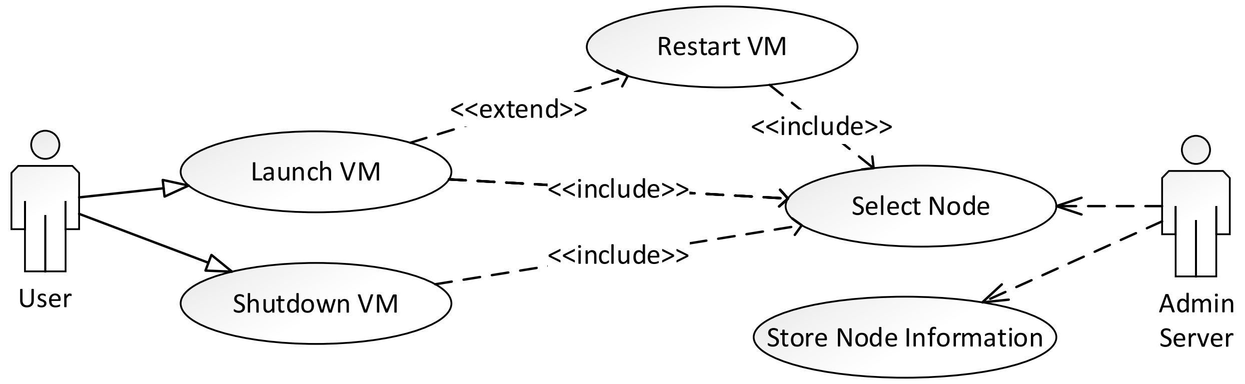


Brief Description:

Users can create VMs with resource (Memory, CPU, Storage) specifications. The admin server selects a Node based on an Algorithm and spins up a (selected OS) VM for the user.

Use case: Start/Stop/Reboot VM

Diagram:

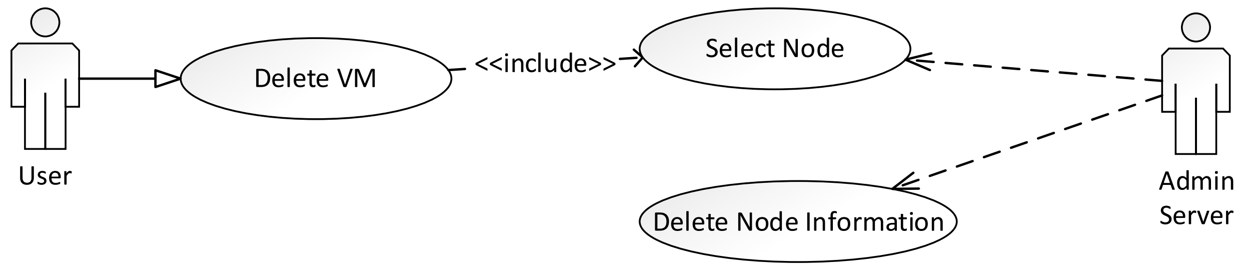


Brief Description:

Users can launch, shutdown, or restart a running VM.

Use case: Delete VM

Diagram:

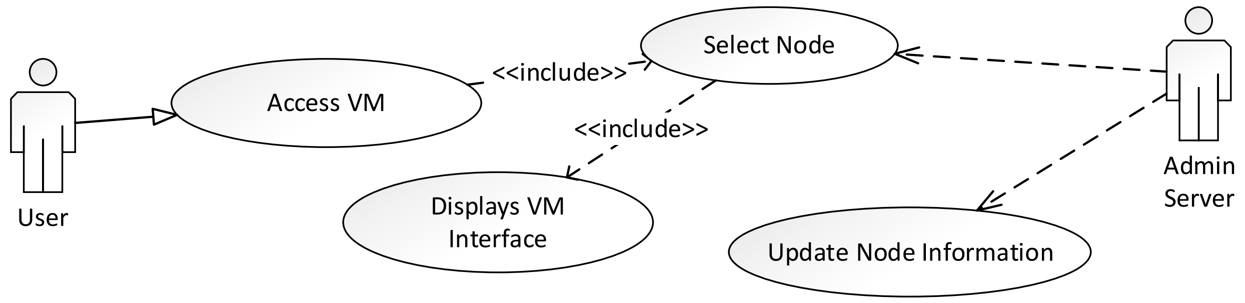


Brief Description:

Users can request to delete an existing Virtual Machine. The admin server to shut down, if running, and remove resources in the VM

Use case: Access VM

Diagram:

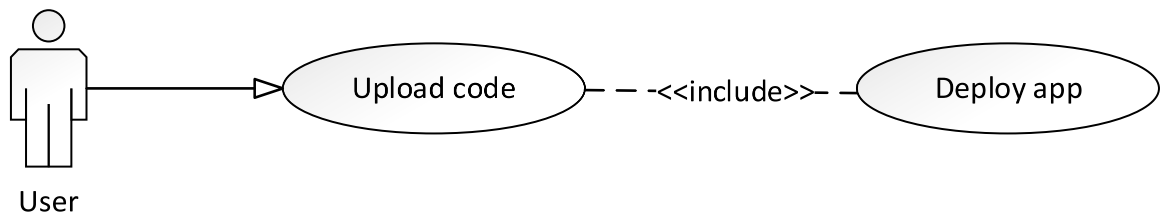


Brief Description:

Users can access the VM through a console provided in the web interface.

Use Case: Upload code

Diagram:



Brief Description:

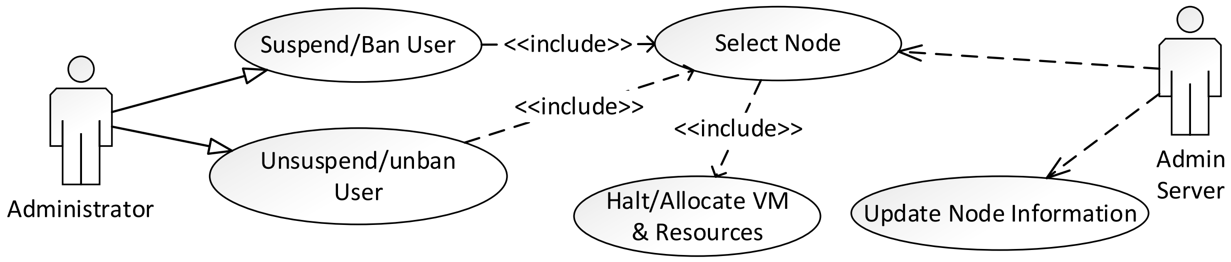
The user can upload code into the VM to deploy apps.

3.1.2 Administrator Use Cases:

Administrators can monitor clusters of virtual machines and manage the performance and resources allocated to each Virtual Machine. Inherits all user use-cases.

Use Case: Manage all users

Diagram:



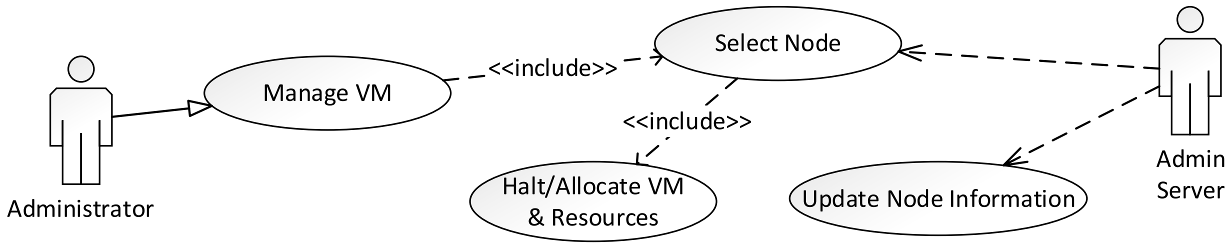
Brief Description:

Administrators can suspend or ban users upon violating user policies.

Administrators can unsuspend or unban users upon compliance with user policies.

Use Case: Manage all VMs

Diagram:

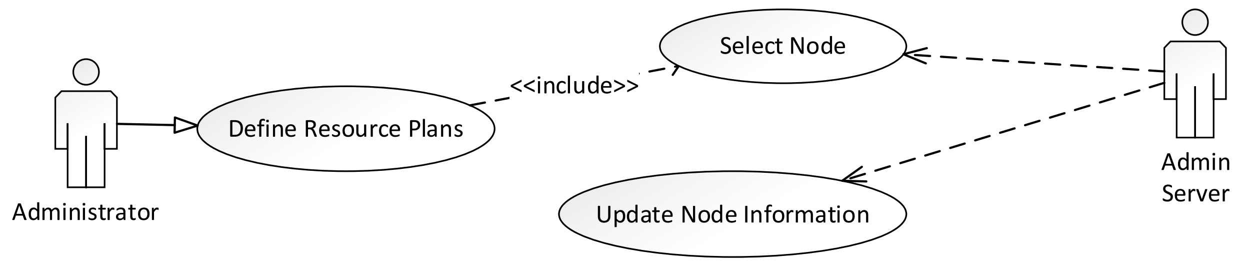


Brief Descriptions:

Admins have control over every VM created on the cluster. In case of an emergency, the admin can gracefully shut down VMs for safety.

Use Case: Define Resource Plans

Diagram:



Brief Description:
Administrators can set the price per resource and compute time of a VM.

3.1.3 Proxmox Cluster Integration & Management

- **API Authentication**: The Admin Server shall securely authenticate with the Proxmox VE API using an API token.
- **Data Synchronization:** The Admin Server shall periodically poll the Proxmox API to synchronize the state of nodes and VMs with its own database.
- **VM State Mapping:** The Admin Server's database shall maintain a mapping: `UserID <-> VmID <-> ProxmoxNodeID`. This is critical for knowing which user owns which VM and where it is located.

3.1.4 Node Selection Algorithm:

When a new VM is requested, the Admin Server shall query all Proxmox nodes and select the one with the most available resources (based on a configurable strategy, Example lowest CPU usage) for placement of VM.

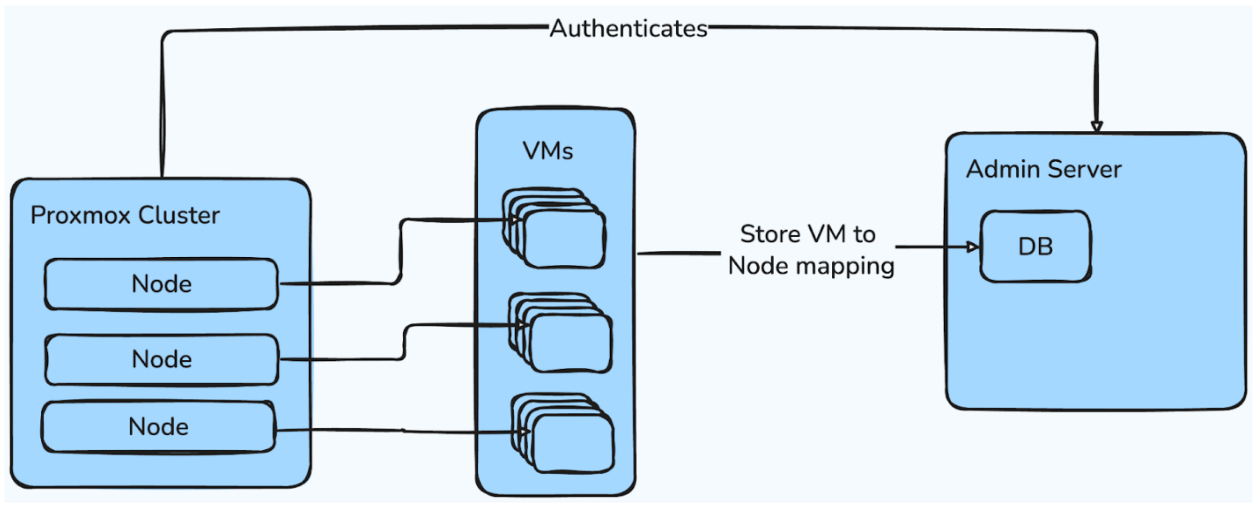


Figure 2: This explains how the admin server maps VMs to the Nodes in the cluster

3.1.5 Resource Allocation and Quotas

- **Resource Plans:** The administrator shall be able to define tiered resource plans ("Basic": 1 vCPU, 2GB RAM, 20GB Disk).

- **User Quotas:** The system shall enforce quotas on a per-user basis (a user can create a maximum of 5 VMs, use a total of 16GB RAM).

4 EXTERNAL INTERFACE REQUIREMENTS

4.1 User Interfaces

- Front-end software: React
- Back-end software: ElysiaJS
- Database software: PostgreSQL

4.2 Hardware Interfaces

Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. Modem, WAN – LAN, Ethernet Cross-Cable.

4.3 Software Interfaces

- A browser that supports CGI, HTML & JavaScript.
- Terminal with SSH capabilities

4.4 Communications Interfaces

- HTTPS: All web traffic shall be over HTTPS.
- Proxmox API: Communication with the Proxmox API shall be over HTTPS (port 8006).

5 NON-FUNCTIONAL REQUIREMENTS

5.1 Performance Requirements

- The product is based on web and has to be run from a web browser on a computer or smart phone.
- The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.

5.2 Security Requirements

- All user passwords shall be hashed and salted using a strong algorithm (e.g., Argon2, bcrypt, SHA256).
- The system shall be protected against common web vulnerabilities, including XSS, CSRF, and SQL Injection.
- Role-Based Access Control (RBAC) shall be strictly enforced. A user must never be able to see or interact with another user's resources.

5.3 Software Requirements

- The servers shall run the latest stable version of Proxmox VE.
- The Admin Server will be a web application running on a dedicated VM or physical server, developed using a modern web stack (Bun / Elysia JS). PostgreSQL is required for the Admin Server.

- The Proxmox nodes must be configured in a cluster with shared storage (Ceph) to enable live migration and High Availability.