

Documentation

Introduction

What is Server Management Suite (SMS)?

Server Management Suite (SMS) is a centralized, secure, and scalable platform built for modern IT infrastructure management. It is designed to simplify the oversight of **Linux** and **Windows** servers across institutions, data centers, and enterprise environments — all from a unified dashboard. SMS empowers system administrators with complete visibility and control over server health, configuration, users, logs, and security alerts.

Whether managing a handful of virtual machines or an entire fleet of servers, SMS reduces complexity by offering modular components that work together seamlessly. It is built with a web-first architecture, allowing remote access, intuitive workflows, and robust automation.

Key Features

- **Real-time Server Health Monitoring** – CPU, memory, disk, network stats
- **Remote Configuration Management** – System/network settings, user roles
- **Token-Based Secure Agent Communication** – Secure controller-to-backend messaging
- **Command Execution** – Send service restarts, file updates, or commands
- **Proactive Alerts and Logging** – Email alerts, audit trails, event logs
- **Modular Architecture** – Easily extensible for routers, firewalls, and third-party tools
- **Cross-Platform Compatibility** – Manage both Windows and Linux hosts

Target Audience

- **University IT Administrators** managing labs and student resource servers
- **Enterprise Infrastructure Teams** overseeing internal server clusters
- **Data Center Operators** who require granular insights and control
- **DevOps Engineers** needing automated, token-secured agent communication
- **SMBs** looking for an open, Docker-ready server management alternative

Getting Started

Prerequisites

Before installation, ensure your system meets the following requirements based on your operating system.

For **Linux Hosts**:

- git
- docker and docker-compose (docker compose v2 CLI)

For **Windows Hosts**:

- Git for Windows [s](#)
- Docker Desktop
- **WSL2** enabled (Windows Subsystem for Linux)
- **Hyper-V** and **Virtualization** enabled in BIOS and Windows Features

Quick Installation Guide

SMS supports **two installation methods** — Docker (recommended) and Manual. This guide uses **Docker**, the fastest way to get up and running.

Docker Deployment (Recommended)

1. Clone the repository

```
git clone https://github.com/kishore-001/ServerManagementSuite.git
cd ServerManagementSuite
```

2. Run the setup script

```
./setup.sh start -> Linux
./setup.ps1 start -> Windows
```

Manual Installation

Ideal for advanced users or those integrating into existing infrastructure.

Clone the Repository

```
git clone https://github.com/kishore-001/ServerManagementSuite.git
cd ServerManagementSuite
```

Frontend Setup

First:

```
cd frontend
npm install
```

Create a `.env` file in `/frontend` with your backend URL:
VITE_BACKEND_URL=http://<your_backend_ip>:9000

Build the frontend:
npm run build

Backend Setup

```
cd backend/db
sqlc generate
cd ..
```

Create a `.env` file in `/backend`:

```
DATABASE_URL=
CLIENT_PORT=
CLIENT_PROTOCOL=
JWT_SECRET=
SERVER_PORT=
LOG_LEVEL=
SMTP_HOST=
SMTP_PORT=
SMTP_USERNAME=
SMTP_PASSWORD=
SMTP_FROM=
```

Then build the backend:

go build -o server [main.go](#)

PostgreSQL Setup

Install PostgreSQL, then run:

sudo systemctl start postgresql

Create a database and user:

CREATE USER admin WITH PASSWORD 'admin';

CREATE DATABASE smsdb;

GRANT ALL PRIVILEGES ON DATABASE smsdb TO admin;

GRANT ALL ON SCHEMA public TO admin;

Confirm DB connection (optional):

psql -U admin -d smsdb -h localhost -p 5432

Initialize the database schema:

go run temp/[dbinit.go](#)

First-Time Login Flow

- Access the frontend via browser
- There will be a default admin account
- Use that to login with password “admin”

Default Username = admin

Default Password = admin

Server Controller Setup

The **Server Controller** is a lightweight agent that runs on every target server you want to manage with SMS. It securely communicates with the central backend, sending real-time health stats, logs, and accepting command/configuration updates.

Collect and report **CPU**, **memory**, **disk**, **network** usage. Stream **logs** and **events** from the server to the SMS backend. Listen for configuration changes or commands (like restart service, push config). Run as a background process or service

Server Setup

Prerequisites

- Go 1.20+
- Git

Installation Steps

Clone the repository

```
git clone https://github.com/kishore-001/ServerManagementSuite.git
cd ServerManagementSuite
```



Build the controller

```
go mod tidy
go build -o controller main.go
```










Run the controller


```
./controller -> Linux
controller.exe -> Windows
```

After controller Start running it will prompt for the token . This will be provided by the Frontend When we add a new server . They will provide a token , we need this token to access the controller safely

```
ServerManagementSuite/client/linux on main [x!?] via  v1.24.5
> ./controller
 Enter token to register this client: █
```

In the above picture we can clearly see that it ask the token for the first time . If you already given the token by mistake you can remove the token by using the below command

```
ServerManagementSuite/client/linux on main [!?] via  v1.24.5
> ls
 api  auth  client.log  controller  go.mod  go.sum  logic  main.go

ServerManagementSuite/client/linux on main [!?] via  v1.24.5
> rm auth/token.hash
```

Command :

```
rm client/linux/auth/token.hash -> Linux
del client\windows\auth\token.hash -> Windows
```

System Architecture

The **Server Management Suite (SMS)** is built with a modular client-server architecture that emphasizes **scalability**, **security**, and **maintainability**. This section outlines the core components, how they communicate, and the typical data flow during runtime.

Core Components

SMS consists of **three main components**, each playing a critical role in system orchestration:

Frontend Web Interface

Tech Stack: React + Vite

Role: Acts as the control center for administrators.

Functions:

- User authentication and role-based access
- Device registration and monitoring
- Viewing logs, alerts, and usage metrics
- Issuing commands and configuration changes

Communicates with: Backend via RESTful API (<http://<backend>:<port>>)

Backend API Server

Tech Stack: Go (Golang)

Role: The central logic and data processing hub.

Functions:

- **API routing for frontend and agent communications**
- **Authentication/token handling**
- **Alert generation and threshold monitoring**
- **Database operations (PostgreSQL)**
- **Command and configuration dispatch**

Ports: Default **80** for frontend, **9000** for API, **2210** for controller agents and **9001** for database

Server Controller Agents

Tech Stack: Go

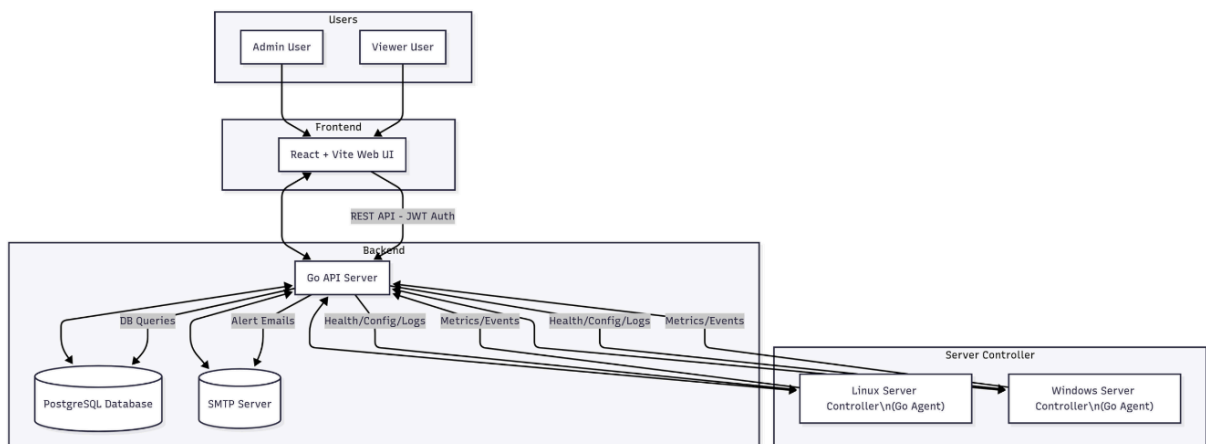
Role: Lightweight agents installed on every managed server.

Functions:

- Periodically report health metrics and logs
- Execute backend-issued commands (restart services, run diagnostics)
- Re-authenticate and reinitialize if token is reset

Authentication: Token-based, generated from frontend

Architecture Diagram



Data Flow Overview

- Admin logs into the frontend dashboard
- Server Controller connects to backend
- Backend processes and stores data
- Frontend pulls live data
- Admin triggers actions

Features & Functionality

The **SMS Admin Dashboard** provides a unified interface to manage your entire server infrastructure — Linux, Windows, and network devices. Below is a detailed explanation of each major module:

Server Monitoring

Monitor the health and performance of all connected servers in real time.

Capabilities:

- View live CPU, memory, disk, and network usage
- System uptime, hostname, OS, and kernel version
- Device online/offline status
- Periodic health snapshots from each Server Controller agent

UI Example:

- Table/Grid view of all servers
- Color-coded health indicators (e.g., green = healthy, red = critical)

Remote Configuration

Admins can update system-level settings across remote servers from a central console.

Capabilities:

- Push configuration files or shell commands to agents
- Restart services or processes (e.g., nginx, sshd)
- Update agent behavior (e.g., reporting interval, thresholds)
- Manage environment variables or system flags remotely

Use Cases:

- Disable SSH root login across multiple machines
- Restart a service in response to a failed health check

Alerts & Notifications

Proactive alerts for critical events and usage anomalies.

Capabilities:

- CPU, Memory, Disk threshold alerts
- Email notifications (via SMTP)
- Alert logging and timestamping

Logs & Metrics

Centralized logging of agent activities and system performance.

Capabilities:

- Agent log streaming to the backend
- View historical metrics for CPU, RAM, disk
- Track executed commands and backend actions
- View agent connection events (connected/disconnected)
- Export logs for forensic or audit analysis (optional extension)

User & Roles Management

Multi-user support with role-based access control (RBAC).

Capabilities:

- Create and manage admin accounts
- Assign roles (e.g., Super Admin, Operator, Read-Only)
- Restrict visibility to specific server groups (planned)
- Secure login with JWT token-based authentication
- Option to reset passwords (future)

Conclusion

The **Server Management Suite (SMS)** is built to address the growing complexity of managing heterogeneous server environments across organizations. With its modular architecture, intuitive dashboard, secure communication protocols, and extensibility, SMS empowers system administrators to take full control of their infrastructure—whether it spans a few systems or thousands. By unifying monitoring, configuration, alerts, and remote control under a single interface, SMS reduces operational overhead and improves reliability across the board.

As the platform continues to evolve, contributions and feedback from the community will drive future enhancements—paving the way for smarter, faster, and more secure server operations.