KVBL Network Backup System - Complete Installation & Feature Documentation (Revised)

1. Overview

This document provides a comprehensive guide for setting up and operating the KVBL Network Backup System. Initially conceived as a standalone unit for individual Network Operations Centers (NOCs), this system has been re-architected into a centralized solution. It now facilitates the management and backup of network device configurations across multiple, geographically dispersed NOCs from a single, central platform. Each NOC maintains its distinct environment, ensuring logical separation while allowing for centralized control and monitoring. Designed for network administrators, this system ensures the reliability, auditability, and disaster recovery readiness of network device configurations within a multi-NOC infrastructure.

Contents

2. Prerequisites

Before commencing the installation, ensure the following tools and services are

provisioned on your system. This section covers prerequisites for both the central backup system and the dedicated jump servers within each Network Operations Center (NOC).

2.1. Software Requirements

The following software components are essential for the KVBL Network Backup System's operation:

Software	Required Version	Notes
Docker	20.10+	Utilized for containerizing the application.
docker-compose	1.29+	Manages multi-container applications.
Google Cloud Platform	(N/A)	Required for generating a Service Account for Google Drive access.
WhatsApp API Server	Custom API	Necessary for sending automated alert notifications.

2.2. Jump Server Prerequisites

Each NOC necessitates a dedicated jump server to establish secure access and execute backup operations on network devices. These jump servers must fulfill the following prerequisites:

- **Python:** A stable Python version (e.g., 3.8+) is mandatory for executing network automation scripts. Python must be installed globally to ensure accessibility from any directory.
- **Netmiko:** The Netmiko library, a multi-vendor SSH connection library, must be installed within the Python environment on each jump server. This library also requires global installation.
  To install Python and Netmiko globally on your jump server, adhere to these steps:
  1. **Update Package Lists:** This command refreshes the list of available packages and their versions from configured repositories.
     sudo apt update

  2. **Install pip for Python 3:** pip is the standard package installer for Python. This command ensures pip is available for installing Python libraries.

sudo apt install python3-pip -y

3. **Check for Netmiko Package Availability:** This command searches for a pre-packaged version of python3-netmiko within your system's repositories.
   apt search python3-netmiko

   If the command yields output similar to the following:
   python3-netmiko/jammy 2.4.2-1 all
     multi-vendor library for SSH connections to network devices - Python 3.X

   Proceed with the repository-based installation:
4. **Install Netmiko (via apt):** If apt search successfully identified a package, this command will install it directly from your system's repositories, ensuring system-wide availability.
   sudo apt update
   sudo apt install python3-netmiko -y

5. **Alternative Netmiko Installation (via pip):** If apt search python3-netmiko does not yield a result, you can install Netmiko using pip, which is the standard Python package installer.
   pip install netmiko

- **fernet_key.txt:** For enhanced SSH connection security, a Fernet key is indispensable. This key must reside in a file named fernet_key.txt within the root directory of each jump server. This key will facilitate the encryption and decryption of sensitive SSH credentials or other data transmitted to/from the jump server and the contents inside the fernet_key.txt should be exactly the same as that of the fernet_key specified in the .env file.
- **Temporary Directory & File Permissions:** The jump server must permit the creation of a temporary directory to receive necessary files from the main server. Specifically, a CSV file (containing device details) and the backup script are transmitted to this temporary directory. Upon completion of the backup operation, a JSON file (containing status updates) is sent back from the jump server to the main server. Appropriate read, write, and execute permissions must be configured for the user account employed by the centralized system to interact with these temporary directories and files on the jump server.

3. Project Setup

3.1. Unzip the Project

Extract the contents of the provided zip archive, KVBL_Network_Backup.zip.

- **For Linux/macOS:**
  unzip KVBL_Network_Backup.zip
  cd KVBL_Network_Backup

- **For Windows:**
  Simply right-click the KVBL_Network_Backup.zip file and select "Extract All..." to a preferred directory. Subsequently, navigate into the extracted KVBL_Network_Backup folder.

4. Installation

4.1. Verify Docker Installation

To ascertain whether Docker and docker-compose are already installed on your system, open a terminal (or PowerShell/Command Prompt on Windows) and execute the following commands:

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

Should these commands return version numbers, you may proceed to the next section. If Docker is not installed, follow the relevant steps below based on your operating system.

4.2. Docker Installation Procedures

4.2.1. Recommended Docker Installation (Ubuntu)

This section outlines a streamlined installation method for Docker and docker-compose on Ubuntu, typically sufficient for most deployment scenarios. In the event of issues with this approach, please refer to the "Alternative Docker Installation" in the subsequent section.

1. **Update Package List:** This action refreshes the list of available packages.
   sudo apt update

2. **Install Docker from Ubuntu's Official Repository:** This command installs the docker.io package, which encompasses the Docker daemon and client.
   sudo apt install -y docker.io

3. **Enable Docker Service on Boot:** This configuration ensures Docker

automatically initiates upon system startup.
sudo systemctl enable docker

4. **Start Docker Service:** This command launches the Docker daemon.
sudo systemctl start docker

5. **Verify Docker Functionality:** Confirm successful installation and operation by checking the Docker version.
docker --version

6. **Add User to docker Group (Optional but Recommended):** This step allows you to execute Docker commands without requiring sudo privileges. A logout and subsequent login are necessary for this change to take effect.
sudo usermod -aG docker $USER

7. **Install docker-compose:** This command downloads and establishes the docker-compose executable.
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

4.2.2. Alternative Docker Installation on Ubuntu (20.04+)

Should the recommended Docker installation method (Section 4.2.1) encounter a failure or if a specific Docker version is required, a more comprehensive installation procedure for Ubuntu (20.04+) is provided below.(in this case use docker compose instead on docker-compose)

Open a terminal and execute the following commands sequentially:

```
sudo apt update
sudo apt install \
   ca-certificates \
   curl \
   gnupg\
   lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

```
echo \
  "deb [arch=$(dpkg --print-architecture) \
  signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
docker --version
sudo docker run hello-world
```

4.2.3. Installing Docker on Windows

For Windows environments, the installation of Docker Desktop is recommended. Docker Desktop bundles Docker Engine, Docker CLI client, docker-compose, Kubernetes, and Credential Helper.

1. Download the installer from the official Docker website: [Download Docker Desktop for Windows]
2. Follow the on-screen instructions provided by the installer.
3. Ensure "WSL 2 backend" is enabled during installation for optimal performance.
4. Post-installation, launch Docker Desktop. You can confirm the installation by opening PowerShell or Command Prompt and executing docker --version and docker-compose --version.

4.2.4. Installing docker-compose Separately (if needed)

If docker-compose is not recognized as a distinct command subsequent to Docker installation (a scenario more prevalent on Linux systems not utilizing Docker Desktop, or with older Docker installations), manual installation is necessary:

```
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

5. Environment Variables

This section details the configuration of environment variables for both the frontend

and backend components of the application.

5.1. frontend/.env Configuration

The frontend/.env file contains configuration specific to the frontend application:

```
# Base URL for the backend API
VITE_API_BASE_URL='/api'
```

5.2. backend/.env Configuration

The backend/.env file houses critical configuration parameters for the backend services:

```
# Fernet encryption key for sensitive data (e.g., SSH credentials)
FERNET_KEY='YOUR_FERNET_KEY_HERE'

# JWT secret key for signing authentication tokens
JWT_SECRET_KEY='YOUR_JWT_SECRET_KEY_HERE'

# MongoDB connection details
MONGO_USERNAME='NetworkBackUp'
MONGO_PASSWORD='NetBackUp123'
MONGO_HOST='mongo'
MONGO_PORT='27017'
MONGO_AUTH_SOURCE='admin'
MONGO_DB='NetworkBackupDB'

# Google Drive shared root folder ID
GOOGLE_DRIVE_SHARED_ROOT_ID='YOUR_GOOGLE_DRIVE_FOLDER_ID'

# WhatsApp API for sending alerts
WHATSAPP_API_URL='http://your.whatsapp.api.url:port/send'
WHATSAPP_API_USER='your_whatsapp_api_user'
WHATSAPP_API_PASSWORD='your_whatsapp_api_password'
```

6. Configuration Steps

6.1. General Configuration

1. The project includes frontend/.env and backend/.env files, populated with placeholder values.
2. Open each file and substitute all placeholder values with your actual keys and credentials.
3. **Crucial Note:** These files must never be committed to version control with real sensitive values.
4. In the absence of .env files, create and populate them manually.

6.2. Google Drive Integration (OAuth 2.0 Method)

To enable cloud-based backup storage using user-based authentication, execute the following steps to configure Google Drive integration. This method uses an OAuth 2.0 Client ID, which requires a user to grant permission through a browser.

6.2.1. Create an OAuth 2.0 Client ID

1. **Access Google Cloud Console**: Navigate to the Google Cloud Console and either select an existing project or create a new one.
2. **Enable Google Drive API**:
    - From the navigation menu, go to **APIs & Services > Library**.
    - Search for "Google Drive API", select it, and click the **Enable** button.
3. **Configure OAuth Consent Screen**:
    - Navigate to **APIs & Services > OAuth consent screen**.
    - Choose **External** for the User Type and click **Create**.
    - Fill in the required application details (App name, User support email, Developer contact information).
    - On the "Scopes" page, you can leave it blank for now.
    - On the "Test users" page, click **+ ADD USERS** and enter the Google email addresses of the users who will be authorized to run the backup service. This must include the email you will use to log in and generate the token in the next steps.
    - Review the summary and click **BACK TO DASHBOARD**.
4. **Create OAuth Client ID**:
    - Navigate to **APIs & Services > Credentials**.
    - Click **+ CREATE CREDENTIALS** and select **OAuth client ID**.
    - For the **Application type**, select **Desktop app**.
    - Give the credential a name (e.g., "Network Backup Desktop Client").
    - Click **Create**.
5. **Download Credentials File**:

- A confirmation window will appear showing your Client ID and Client Secret. Click **DOWNLOAD JSON**.
- Rename the downloaded file to `client_secrets.json`.
- Place this `client_secrets.json` file inside the `/KVBL_Network_Backup/backend/` directory.

6.2.2. Generate Authentication Token

To authorize the application to access user data, you need to run a Python script to generate a `token.json` file.

1. **Prerequisites**: Ensure you have the necessary Google client libraries installed for Python. If not, open your terminal and run:
   pip install google-api-python-client google-auth-httplib2 google-auth-oauthlib

   Run `generate_token.py`: Run `generate_token.py` in the same directory where you placed `client_secrets.json` (i.e., the `backend` folder).

2. **Run the Script**: Navigate to the `backend` directory in your terminal and run the script:
   python generate_token.py
   1. **Authorize Access**:
      - The script will print a URL in the console. Copy this URL and paste it into your web browser.
      - Log in with the Google account you added as a "Test user" in the consent screen configuration.
      - Grant the application permission to access your Google Drive.
      - After successful authorization, you will be redirected to a page that may show a "This site can't be reached" error. This is expected behavior as the script is running a temporary local server just to capture the token. You can close the browser tab.
   2. **Place Token File**: The script will create a `token.json` file in the `backend` directory. This file contains the access and refresh tokens your application needs. **Do not move this file.**

6.2.3. Set Up Google Drive Backup Folder & Update Environment

1. **Create Backup Folder**: In your Google Drive, create a new folder that will serve as the repository for backups (e.g., `Network_Device_Backups`).
2. **Obtain Folder ID**: Open the folder in your browser. The Folder ID is the last

part of the URL.
    - Example URL: `https://drive.google.com/drive/folders/1a2b3c4d5e6f7g8h9i0j`
    - Your Folder ID: `1a2b3c4d5e6f7g8h9i0j`
3. **Update Environment Variable**: Paste this Folder ID into your `backend/.env` file for the `GOOGLE_DRIVE_SHARED_ROOT_ID` variable: GOOGLE_DRIVE_SHARED_ROOT_ID='YOUR_FOLDER_ID'

Important Note: The backend application logic will need to be updated to use the `client_secrets.json` and `token.json` files for authentication instead of the previous service account method.

7. Building the Application

The application leverages docker-compose to orchestrate its services: a frontend (React), a backend (FastAPI), and a MongoDB database.

7.1. Service Port Mappings

The application's services are mapped to specific ports for internal and external access:

- **Frontend Service (Nginx):**
    - Exposes port 80 internally within the container.
    - Mapped to port 5000 on your host machine.
    - The frontend is accessible at: http://localhost:5000
- **MongoDB Service:**
    - Exposes port 27017 internally within the container.
    - Mapped to port 28017 on your host machine.
    - Allows external tools to connect to MongoDB at: localhost:28017
- **Backend Service:**
    - Exposes port 8000 internally within the Docker network.
    - This port is not directly mapped to the host, as frontend requests are proxied through Nginx.

7.2. Building Docker Images

Navigate to the project's root directory (where docker-compose.yml is located) and execute the following command to build the Docker images for all services:

```
docker-compose build
```

Note: On newer Ubuntu versions (22.04+), use docker-compose (hyphenated) instead of docker compose (space-separated). If permission errors are encountered, prefix the command with sudo.

To build images without leveraging cache, use the following command:

```
docker-compose build --no-cache
```

8. Starting the Application

The application services can be initiated either in the foreground for real-time log observation or in the background for continuous operation.

8.1. Starting in Foreground

To start the services in the foreground:

```
docker-compose up
```

8.2. Starting in Background (Detached Mode)

To initiate the services in the background (detached mode), employ the -d flag:

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

8.3. Combined Build and Start Command

For convenience, the build and start operations can be combined into a single command:

```
docker-compose up --build -d
```

9. Stopping Services

To terminate the running services:

```
docker-compose down
```

To stop services and concurrently remove named volumes (such as MongoDB data), use:

docker-compose down --volumes

10. Accessing the Application

The frontend of the application is accessible via your web browser at: http://localhost:5000

The backend API operates internally within the Docker network on port 8000.

11. Important Security Note: Default Head Admin

Upon initial system startup, a default Head Admin account is provisioned with the following credentials:

- **Username:** head_admin
- **Password:** head_admin1234

Action Required: It is imperative to log in immediately upon first access, change the default password, and create a new, secure administrator account. For security reasons, the default head_admin account (or any Head Admin) cannot be deleted directly through the application's user interface. Should the removal of a Head Admin account be necessary, it must be performed directly from the database.

Procedure for Deleting a Head Admin Directly from the Database:

1. **Access the MongoDB Container:** Utilize the docker exec command to establish a mongosh shell session inside your MongoDB container. Replace <container_name>, <username>, <password>, and <auth_db> with your specific MongoDB container details and access credentials.
   docker exec -it <container_name> mongosh -u <username> -p <password> --authenticationDatabase <auth_db>

   Example:
   docker exec -it netbackup-copy-networkmongo-1 mongosh -u NetworkBackUp -p NetBackUp123 --authenticationDatabase admin

2. **Display Databases:** Once within the mongosh shell, you can list the available

databases to identify your application's database.
show dbs

Example Output:
NetworkBackupDB 944.00 KiB
admin 100.00 KiB
config 96.00 KiB
local 128.00 KiB

3. **Switch to Your Database:** Employ the use command to switch to your application's database (e.g., NetworkBackupDB).
use <database_name>

Example:
use NetworkBackupDB

4. **Delete the Head Admin User:** Execute the deleteOne command on the LoginInfo collection to remove the specified head admin user. If the username is different, replace "head_admin" with the correct username.
db.LoginInfo.deleteOne({ username: "head_admin" })

12. NOC Management

Following successful authentication as the Head Admin, it is paramount to proceed with the immediate setup of all requisite Network Operations Centers (NOCs). This action establishes the fundamental framework for managing network devices across diverse geographical locations or logical segments.

To register a new NOC, navigate to the dedicated NOC Management section (typically accessible via the sidebar or within Account Settings). A form will be presented, prompting for the following details for each NOC:

Field	Example Value	Description
NOC Name	East Client NOC	A unique identifier for the Network Operations Center.
Description (Optional)	NOC for Eastern Region	A brief explanation detailing the NOC's purpose or geographical coverage.

Contact Information (Optional)		
Location	New York, USA	The physical address of the NOC.
Contact Person	John Doe	The primary individual responsible for this NOC.
Contact Email	j.doe@example.com	The email address for the designated contact person.
Contact Phone	1234567890	The telephone number for the designated contact person.
NOC Connectivity (Jump Server) Configuration		Essential details for the central system to establish connection with the jump server within this NOC's environment.
NOC/Jump Server Hostname/IP	192.168.1.1 or jump.yourdomain.com	The IP address or hostname of the jump server.
Username for the NOC	noc_user	The username used for accessing the jump server for this specific NOC.
Password (Optional, if using Private Key)	secure_password	The password for password-based authentication to the jump server.
Confirm Password	secure_password	Re-enter the password for confirmation.
SSH Private Key (Optional, if using Password)	-----BEGIN OPENSSH PRIVATE KEY-----...-----END OPENSSH PRIVATE KEY-----	The SSH private key content for key-based authentication to the jump server. Must include full header/footer.
NOC Port	22	The port number utilized for SSH connections to the jump server.

Upon completion of all required information, click Create NOC to integrate the new Network Operations Center into the system. This process should be reiterated for

every necessary NOC within your infrastructure.

13. Troubleshooting

This section provides guidance on diagnosing and resolving common issues encountered during the application's operation.

13.1. Container Exit with Error Code

To inspect logs for a specific service and identify the cause of an error:

docker-compose logs <service_name>

13.2. Port Conflicts

To identify which process is currently utilizing a specific port:

sudo lsof -i :<port_number>

13.3. Network Issues

To diagnose network-related problems within the Docker environment:

- **List Docker Networks:**
  docker network ls

- **Inspect a Specific Network:**
  docker network inspect <network_name>

13.4. Volume Permission Issues

If difficulties arise with MongoDB data volume permissions (e.g., /data/db), it may be necessary to adjust ownership:

chown -R appuser:appuser /data/db

13.5. Common Docker Commands

The following table presents frequently used Docker commands for managing and troubleshooting containers and images:

Command	Description
docker ps	Lists all currently running containers. (docker ps -a shows all containers, including stopped ones.)
docker logs <container_name_or_id>	Displays the logs for a specified container. (Add -f to follow logs in real-time.)
docker inspect <container_name_or_id>	Provides comprehensive details about a container, including network settings and volumes.
docker images	Lists all Docker images present on the system.
docker stop <container_name_or_id>	Halts a running container.
docker rm <container_name_or_id>	Removes a stopped container.
docker rmi <image_name_or_id>	Deletes a Docker image from the system.

14. Security Recommendations 🔒

Following the initial setup and the secure creation of the Head Admin account, adherence to the subsequent security recommendations is advised:

1. Remove Default Admin Creation Code:
   Locate and remove the following lines of code from backend/main.py, which are responsible for the default head admin provisioning:
   from test import create_head_admin_if_missing
   create_head_admin_if_missing() # Ensure head_admin user exists

2. Delete test.py:
   Remove the backend/test.py file if it contains any sensitive information or default configurations.

14.1. Recommended Secure Startup Sequence

1. Initiate the application services:
   docker-compose up -d

2. Log in as the head_admin user.
3. Create a new, robust Head Admin account.
4. Eliminate the default admin creation logic from backend/main.py.

5. Delete the backend/test.py file.
6. Restart services to apply the implemented changes:
   docker-compose down
   docker-compose up -d

15. Navigation Guide & System Access 🌐

This section offers a visual guide to the KVBL Network Backup System's user interface elements and navigation pathways.

15.1. Login Page 🔐

A secure login form facilitates user authentication by validating credentials against bcrypt-hashed passwords. It accommodates both Admin and general User roles.

15.2. Sidebar Navigation ➡️

The sidebar serves as a primary navigation hub, providing quick access to various application sections:

- Dashboard
- Devices
- Phone Numbers
- Backup Schedule
- Compare Files
- Backups
- Account Settings

15.3. Navbar Components 🔝

The top navigation bar provides access to essential features and user preferences:

- Recent Actions
- Current User
- Preferences
- Switching NOCs

16. Core Features & Functionality ⭐

This section delineates the comprehensive features offered by the KVBL Network Backup System, now enhanced for centralized multi-NOC management capabilities.

16.1. Dashboard Details 📊

The dashboard provides real-time system monitoring across all managed NOCs,

displaying key operational metrics:

- Bar graph illustrating active and inactive devices.
- Donut graph categorizing devices by type.
- Total number of registered phone numbers.
- Total count of managed devices.
- A quick overview detailing device backup status, active status, and IP addresses.
- Information regarding the next scheduled backup.
- Recent logs and system activities.

16.2. Device Management 🧠

- **Device Administration:** Facilitates the addition, editing, and removal of network devices.
- **Prerequisite:** Prior to adding a device, a corresponding device type must be defined and established within the system.
- **Condition:** While adding a new device make sure that the password is in visible mode.
- **Device Type Definition:** Allows for the definition of various device types (e.g., Huawei, Arista EOS).
- **Command Template Assignment:** Enables the assignment of specific command templates per device type.
- **Ping Testing:** Supports both scheduled and manual ping tests for device connectivity verification.
- **Backup Execution via Jump Server:** When a backup operation is initiated for a device, the centralized system first establishes a secure connection to the specified jump server within the corresponding NOC's environment. The backup function is subsequently executed by the jump server, which interacts directly with the target network device. The centralized system transmits a CSV file (containing device details) and the backup script to a temporary directory created on the jump server. Upon completion of the backup operation, a JSON file containing the status update is sent back from the jump server to the main server. This process mandates the configuration of appropriate file transfer permissions on the jump server.

16.3. Phone Number Management 📞

- **Notification Management:** Enables the addition of phone numbers designated for receiving notifications.
- **Preference Configuration:** Allows users to set preferences for receiving device alerts, backup alerts, or both.
- **Secure Storage:** Ensures the secure storage of phone number data.

16.4. Scheduled Backups ⏰

- **Automated Scheduling:** Permits the scheduling of automated backups across all managed NOCs.
- **Frequency Options:** Supports selection of backup frequency: Daily, Weekly, or Monthly.
- **Custom Time Configuration:** Provides granular control over custom time configuration (hour + minute).
- **Minimum Scheduling Delay:** Users are required to schedule backups with a minimum delay of 30 minutes from the current time. For instance, if the current time is 1:00 PM, the earliest permissible backup time would be 1:30 PM. This validation is integrated into the backend logic, specifically within the schedule_service.py file, to enforce time constraints before new or updated scheduled jobs are processed.
- Implementation Detail (backend/services/schedule_service.py):
  The add_scheduled_backup_jobs function in backend/schedule_service.py is responsible for refreshing and updating the scheduled backup jobs. The minutes=60 parameter specifies that the scheduler checks for updates every 60 minutes. This interval can be adjusted (e.g., to seconds, minutes, hours) as required for more frequent checks. Any modifications to the minimum scheduling delay or the job refresh frequency would be configured within this file.

```
# File path: backend/schedule_service.py
# ... other imports and code ...
scheduler.add_job(
    add_scheduled_backup_jobs, # This function re-reads from DB and updates jobs
    'interval',
    minutes=60, # Check every 60 minutes (can be changed to hours, seconds, etc.)
    id='refresh_backup_schedules_job',
    replace_existing=True,
    timezone=ist # Ensure it adheres to IST
)
# ... rest of the code ...
```

16.5. Compare Backups 🗒️

- **Version Comparison:** Enables the fetching and comparison of different backup versions.
- **Configuration Change Identification:** Utilizes hashing algorithms to identify

configuration changes between versions.
- **Visual Difference Display:** Provides a visual representation of differences between configuration files.
- **Historical Version Download:** Supports downloading historical versions in ZIP format or as individual files.
- **Access Restriction:** Regular users are restricted to comparing files only from the NOCs to which they are assigned. Head Admins possess unrestricted access and can compare any files across all NOCs.

16.6. Recent Backups

- **Backup Status Overview:** Displays the backup status for all devices, including their current status and last backup status across all NOCs.
- **Manual Backup Trigger:** Offers an option to manually trigger backups for all or a selected number of devices.
- **Data Management:** Includes refresh and filter options for managing backup data.

16.7. Recent Actions 📜

- **Event Logging:** Records system events and user actions performed by any user or the system itself across all NOCs.
- **Examples:** Device addition/removal, backup execution.

16.8. Preferences ⚙️

This section empowers users to personalize their interface experience.

- **Theme Options:**
  - Dark Mode
  - Light Mode
- **Display Options:**
  - Compact Mode
  - Hide Sidebar

16.9. Account Settings 🔑

This section provides authorized users with the capability to manage user accounts and system settings, featuring enhanced functionalities for multi-NOC management.

- 16.9.1. NOC Access Management
  This functionality enables Head Admins to meticulously control which NOCs specific Administrators and Users can access.
  - **Head Admin Privilege:** The Head Admin possesses inherent and automatic access to manage all NOCs within the centralized system. This is a default,

non-configurable privilege.
    - **Assigning NOCs to Administrators and Users:** Head Admins can grant or revoke access to specific NOC environments for individual Admin and User accounts. This mechanism ensures that only authorized personnel can view and manage devices or perform operations within designated NOCs.
    - **Multiple NOC Access:** Both Administrators and Users can be assigned access to multiple NOCs, facilitating the management of devices and operations across their designated areas of responsibility from a single login.
- **Administrator and Head Administrator Privileges:** Users with these roles can view and modify information pertaining to other users.
- **Head Administrator Exclusive Privileges:**
    - Promote or demote user roles.
    - Activate or suspend user accounts.
    - Delete and add new users (including other Head Admins and regular Admins). **Note:** Head Admins cannot be deleted directly from the user interface. Refer to Section 11 for detailed instructions on deleting a Head Admin directly from the database.
    - Automatically assigned control over all existing and newly added NOCs.
- **Administrator Exclusive Privileges:**
    - Add or delete regular users.
- **User Information:** The settings display comprehensive user details such as email address, full name, username, last login IP, and additional pertinent information.