Raspberry Pi Monitoring Bot: Installation and Setup Guide

This document provides a detailed, step-by-step guide for deploying the Monitoring Bot Suite on a Raspberry Pi. It covers all necessary steps, from cloning the repository to setting up the systemd services to ensure all components run automatically and reliably on boot.

1. Prerequisites

Before you begin, ensure your Raspberry Pi is running a recent version of Raspberry Pi OS (formerly Raspbian). You must have git and python3 with venv installed.

- **Python 3:** The scripts are written in Python 3.
- **Git:** Required to clone the project repository.

2. Directory Structure

The project has a clear directory structure that organizes the scripts, logs, and reports. All paths are relative to the project's root directory: /home/adm_onkar/Monitoring_script.

```
/home/adm_onkar/Monitoring_script/
├──.env                # Environment file for bot token (ignored by git)
├──.gitignore          # Git ignore file for logs, venvs, etc.
├── bot.py             # Main Telegram bot script
├── bot_debug.log      # Log file for the bot script
├── command_filter.py  # Whitelist for safe shell commands
├── logs/              # Directory for the database and log files
```

```
│   └── monitoring_data.db  # SQLite database for system and network metrics
├── network_monitor.log # Log file for network monitoring
├── system_monitor.log  # Log file for system resource monitoring
├── module1.py          # Network connectivity monitoring script
├── monitoring_venv/    # Python virtual environment for dependencies
├── report/             # Directory for generated PDF reports
│   └── *.pdf           # Example generated report
├── report_generator.py   # Script to generate PDF reports from database data
├── start_monitoring.sh   # Bash script to activate venv and run the bot
└── system_monitor.py     # System resource monitoring script
```

3. Clone the Repository

First, log into your Raspberry Pi via SSH or directly and clone the project repository to your desired directory. The user's systemd service files indicate the working directory is /home/adm_onkar/Monitoring_script.

Bash

```
git clone https://github.com/onkarautade/Monitoring_bot.git /home/adm_onkar/Monitoring_script
cd /home/adm_onkar/Monitoring_script
```

4. Set up the Virtual Environment and Dependencies

According to the provided startup script, bot.py must run within a dedicated virtual environment. This is a best practice that isolates the project's dependencies from the global Python installation.

1. **Create the virtual environment:**
   Bash
   ```
   python3 -m venv monitoring_venv
   ```

2. **Activate the environment:**
   Bash
   ```
   source monitoring_venv/bin/activate
   ```

3. **Install dependencies:** The requirements.txt file contains all the necessary libraries.
   Bash
   ```
   pip install -r requirements.txt
   ```

5. Configure the Bot

The bot requires a Telegram bot token to operate. This token must be stored in a file named .env in the project's root directory.

1. **Create the .env file:**
   Bash
   ```
   nano.env
   ```

2. **Add your token to the file:** Replace YOUR_BOT_TOKEN_HERE with your actual Telegram bot token.
   Ini, TOML
   ```
   BOT_TOKEN=YOUR_BOT_TOKEN_HERE
   ```

3. **Update bot_clean.py:** You must also modify bot_clean.py to add your Telegram Chat ID to the ALLOWED_CHAT_ID set. This is crucial for granting access to the bot.[1]

6. Create Necessary Directories and Database

The scripts log data and generate reports, which require specific directories and a database file.

1. **Create the logs and report directories:** The system_monitor.py [1] and module1.py [1] scripts store log files and a database in the logs directory. The report_generator.py script [1] saves PDFs to the report directory.
   Bash
   ```
   mkdir -p logs report
   ```

2. **Initialize the database file:** The system_monitor.py [1] and module1.py [1] scripts will automatically create the necessary tables, but the database file itself needs to exist.
   Bash
   ```
   touch logs/monitoring_data.db
   ```

7. Configure systemd Services for Autostart

The provided systemd unit files will ensure the core monitoring processes start automatically at boot and restart if they fail.

1. **Create the systemmonitor.service file:** This service runs the system_monitor.py script [1], which continuously logs system metrics to the database.
   Bash
   ```
   sudo nano /etc/systemd/system/systemmonitor.service
   ```

   File Content:
   Ini, TOML
   ```
   [Unit]
   Description=system monitor
   After=network.target syslog.target local-fs.target
   Requires=network-online.target
   Wants=network-online.target


   Type=simple
   User=adm_onkar
   WorkingDirectory=/home/adm_onkar/Monitoring_script
   ExecStart=/usr/bin/python3 /home/adm_onkar/Monitoring_script/system_monitor.py
   Restart=always
   RestartSec=10
   StandardOutput=syslog
   StandardError=syslog
   SyslogIdentifier=systemmonitor
   Environment=PATH=/usr/bin:/usr/local/bin
   KillMode=process
   TimeoutStartSec=300
   ```

```
[Install]
WantedBy=multi-user.target
```

2. **Create the module1.service file:** This service runs the module1.py script [1], which monitors network connectivity and logs the data.
   Bash
   ```
   sudo nano /etc/systemd/system/module1.service
   ```

   File Content:
   Ini, TOML
   ```
   [Unit]
   Description=Module1 Network Monitoring
   After=network.target


   Type=simple
   User=adm_onkar
   WorkingDirectory=/home/adm_onkar/Monitoring_script
   ExecStart=/usr/bin/python3 /home/adm_onkar/Monitoring_script/module1.py
   Restart=always
   RestartSec=10

   [Install]
   WantedBy=multi-user.target
   ```

3. **Create the start_monitoring.sh script:** This script is necessary to activate the virtual environment before running the bot.
   Bash
   ```
   nano /home/adm_onkar/Monitoring_script/start_monitoring.sh
   ```

   File Content:
   Bash
   ```
   #!/bin/bash
   cd /home/adm_onkar/Monitoring_script
   source monitoring_venv/bin/activate
   python3 bot.py
   ```

   Set execute permissions for the script:
   Bash
   ```
   chmod +x /home/adm_onkar/Monitoring_script/start_monitoring.sh
   ```

4. **Create the monitoring_bot.service file:** This service runs the start_monitoring.sh script, which launches the main Telegram bot [1] in the virtual environment.

Bash
```
sudo nano /etc/systemd/system/monitoring_bot.service
```

File Content:
Ini, TOML
```
[Unit]
Description=Monitoring Bot Service
After=network.target


User=adm_onkar
WorkingDirectory=/home/adm_onkar/Monitoring_script
ExecStart=/home/adm_onkar/Monitoring_script/start_monitoring.sh
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

8. Start and Enable the Services

With the service files created, you can now tell systemd to recognize and run them.

1. **Reload the systemd daemon to pick up the new service files:**
   Bash
   ```
   sudo systemctl daemon-reload
   ```

2. **Start all three services:**
   Bash
   ```
   sudo systemctl start systemmonitor.service
   sudo systemctl start module1.service
   sudo systemctl start monitoring_bot.service
   ```

3. **Enable the services to start automatically on boot:**
   Bash
   ```
   sudo systemctl enable systemmonitor.service
   sudo systemctl enable module1.service
   sudo systemctl enable monitoring_bot.service
   ```

9. Verify Installation

You can confirm that the services are running and working correctly by checking their status and logs.

- **Check service status:**
  Bash
  ```
  sudo systemctl status systemmonitor.service
  sudo systemctl status module1.service
  sudo systemctl status monitoring_bot.service
  ```

- **Check logs for issues:**
  Bash
  ```
  sudo journalctl -u systemmonitor.service -f
  sudo journalctl -u module1.service -f
  sudo journalctl -u monitoring_bot.service -f
  ```

After a few minutes, you can also send the /start command to your Telegram bot to test its functionality.