Migration Guide: Windows to RHEL

System Requirements

RHEL Setup

1. RHEL 8 or later
2. Minimum specifications:
    - 4GB RAM
    - 20GB disk space
    - 2 CPU cores

Required Packages

```
# Update system
sudo dnf update -y

# Install required packages
sudo dnf install -y nodejs npm postgresql-server postgresql-contrib nginx git

# Install PM2 globally
sudo npm install -g pm2
```

PostgreSQL Setup

1. Initialize PostgreSQL

```
# Initialize PostgreSQL database
sudo postgresql-setup --initdb

# Start and enable PostgreSQL
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

2. Configure PostgreSQL

```
# Edit PostgreSQL configuration
sudo vi /var/lib/pgsql/data/postgresql.conf

# Add/modify these lines:
listen_addresses = '*'
max_connections = 100
shared_buffers = 128MB
```

3. Configure Access

```
# Edit pg_hba.conf
sudo vi /var/lib/pgsql/data/pg_hba.conf

# Add these lines:
host    all             all             0.0.0.0/0               md5
host    all             all             ::/0                    md5
```

4. Create Master Database

```
-- Connect to PostgreSQL
psql -U postgres

-- Create master_db
CREATE DATABASE master_db;
\c master_db

-- Create sequences and tables
-- Users Table
CREATE SEQUENCE IF NOT EXISTS public.users_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

CREATE TABLE IF NOT EXISTS public.users (
    id integer NOT NULL DEFAULT nextval('users_id_seq'::regclass),
    username character varying(255) NOT NULL,
    api_key_hash text,
    api_key_plain text,
    created_at timestamp without time zone DEFAULT now(),
    is_active boolean DEFAULT true,
    CONSTRAINT users_pkey PRIMARY KEY (id),
    CONSTRAINT users_username_key UNIQUE (username)
);

-- Credentials Table
CREATE SEQUENCE IF NOT EXISTS public.credentials_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

CREATE TABLE IF NOT EXISTS public.credentials (
    id integer NOT NULL DEFAULT nextval('credentials_id_seq'::regclass),
```

```
    user_id integer NOT NULL,
    key_name character varying(255),
    key_value text,
    created_at timestamp without time zone DEFAULT now(),
    username character varying(255),
    CONSTRAINT credentials_pkey PRIMARY KEY (id),
    CONSTRAINT credentials_user_id_fkey FOREIGN KEY (user_id)
        REFERENCES public.users (id) ON DELETE CASCADE
);

-- Dashboard Tables
CREATE SEQUENCE IF NOT EXISTS public.dashboardtiming_id_seq AS integer START WITH
1;
CREATE SEQUENCE IF NOT EXISTS public.dashboardqor_id_seq AS integer START WITH 1;
CREATE SEQUENCE IF NOT EXISTS public.dashboarddrc_id_seq AS integer START WITH 1;

-- Timing Dashboard Table
CREATE TABLE IF NOT EXISTS public.dashboardtiming (
    id integer NOT NULL DEFAULT nextval('dashboardtiming_id_seq'::regclass),
    user_id integer,
    username text,
    table_name text,
    dashboard_url text,
    local_snapshot_url text,
    source text DEFAULT 'grafana'::text,
    created_at timestamp with time zone DEFAULT now(),
    app_user_id integer,
    app_username character varying(255),
    slack_sent_at timestamp with time zone,
    CONSTRAINT dashboardtiming_pkey PRIMARY KEY (id),
    CONSTRAINT fk_app_user_timing FOREIGN KEY (app_user_id)
        REFERENCES public.users (id)
);

-- QOR Dashboard Table
CREATE TABLE IF NOT EXISTS public.dashboardqor (
    id integer NOT NULL DEFAULT nextval('dashboardqor_id_seq'::regclass),
    user_id integer,
    username text,
    table_name text,
    dashboard_url text,
    local_snapshot_url text,
    source text DEFAULT 'grafana'::text,
    created_at timestamp with time zone DEFAULT now(),
    app_user_id integer,
    app_username character varying(255),
    slack_sent_at timestamp with time zone,
    CONSTRAINT dashboardqor_pkey PRIMARY KEY (id),
    CONSTRAINT fk_app_user_qor FOREIGN KEY (app_user_id)
        REFERENCES public.users (id)
);

-- DRC Dashboard Table
CREATE TABLE IF NOT EXISTS public.dashboarddrc (
```

```
    id integer NOT NULL DEFAULT nextval('dashboarddrc_id_seq'::regclass),
    user_id integer,
    username text,
    table_name text,
    dashboard_url text,
    local_snapshot_url text,
    source text DEFAULT 'grafana'::text,
    created_at timestamp with time zone DEFAULT now(),
    app_user_id integer,
    app_username character varying(255),
    slack_sent_at timestamp with time zone,
    CONSTRAINT dashboarddrc_pkey PRIMARY KEY (id),
    CONSTRAINT fk_app_user_drc FOREIGN KEY (app_user_id)
        REFERENCES public.users (id)
);

-- Set permissions
ALTER TABLE public.users OWNER TO postgres;
ALTER TABLE public.credentials OWNER TO postgres;
ALTER TABLE public.dashboardtiming OWNER TO postgres;
ALTER TABLE public.dashboardqor OWNER TO postgres;
ALTER TABLE public.dashboarddrc OWNER TO postgres;
```

Grafana Setup

1. Install Grafana

```
# Add Grafana repository
sudo tee /etc/yum.repos.d/grafana.repo << EOF
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF

# Install Grafana
sudo dnf install -y grafana

# Start and enable Grafana
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

2. Configure Grafana

```
# Edit Grafana configuration
sudo vi /etc/grafana/grafana.ini

[server]
protocol = http
http_addr = 0.0.0.0
http_port = 3000

[security]
allow_embedding = true
cookie_secure = false

[auth.anonymous]
enabled = true
```

Application Setup

1. Directory Structure

```
# Create application directory
sudo mkdir -p /opt/credential-grafana
sudo chown -R $USER:$USER /opt/credential-grafana

# Create GIF capture directory
sudo mkdir -p /opt/credential-grafana/captureGrafanaGIF
sudo chmod 755 /opt/credential-grafana/captureGrafanaGIF
```

2. Environment Configuration

```
# Create environment file
sudo vi /opt/credential-grafana/server_side/.env

# Server Configuration
PORT=8050
HOST=0.0.0.0
FRONTEND_URL=http://YOUR_IP:8051

# Database Configuration
MASTER_DB_HOST=localhost
MASTER_DB_PORT=5432
MASTER_DB_USER=postgres
MASTER_DB_PASS=your_password
MASTER_DB_NAME=master_db

# Grafana Configuration
GRAFANA_BASE_URL=http://localhost:3000
GRAFANA_USER=admin
GRAFANA_PASSWORD=your_grafana_password
```

```
# GIF Configuration
GIF_CAPTURE_DIR=/opt/credential-grafana/captureGrafanaGIF
CAPTURE_WIDTH=1920
CAPTURE_HEIGHT=1080
CAPTURE_FRAMES=10
CAPTURE_FRAME_DELAY=500
CAPTURE_TIMEOUT=120000
```

3. SELinux Configuration

```
# Allow Node.js to make network connections
sudo setsebool -P httpd_can_network_connect 1

# Allow Node.js to write to GIF directory
sudo semanage fcontext -a -t httpd_sys_rw_content_t "/opt/credential-
grafana/captureGrafanaGIF(/.*)?"
sudo restorecon -Rv /opt/credential-grafana/captureGrafanaGIF
```

4. Firewall Configuration

```
# Open required ports
sudo firewall-cmd --permanent --add-port=8050/tcp
sudo firewall-cmd --permanent --add-port=8051/tcp
sudo firewall-cmd --permanent --add-port=3000/tcp
sudo firewall-cmd --permanent --add-port=5432/tcp
sudo firewall-cmd --reload
```

Data Migration

1. Export Existing Data

```
# On Windows system
pg_dump -U postgres master_db > master_db_backup.sql
```

2. Import Data to RHEL

```
# Copy backup to RHEL system
scp master_db_backup.sql user@rhel_server:/tmp/

# On RHEL system
psql -U postgres master_db < /tmp/master_db_backup.sql
```

Application Deployment

1. Clone Repository

```
cd /opt/credential-grafana
git clone [repository-url] .
```

2. Install Dependencies

```
# Backend
cd server_side
npm install

# Frontend
cd ../frontend
npm install
```

3. Start Application

```
# Start backend
cd /opt/credential-grafana/server_side
pm2 start index.js --name credential-grafana-backend

# Start frontend
cd /opt/credential-grafana/frontend
pm2 start npm --name credential-grafana-frontend -- start

# Save PM2 configuration
pm2 save
```

Post-Migration Checks

1. Database Verification

```
-- Check tables
\dt

-- Check sequences
\ds

-- Verify permissions
\du
```

2. Application Checks

1. Verify frontend access: `http://YOUR_IP:8051`
2. Test backend API: `http://YOUR_IP:8050/api/v1/users`
3. Confirm Grafana access: `http://YOUR_IP:3000`
4. Test GIF capture functionality
5. Verify Slack integration

3. Monitoring Setup

```
# Monitor logs
pm2 logs

# Monitor system resources
top
df -h
free -m
```

Troubleshooting

Common Issues

1. **PostgreSQL Connection Issues**

```
# Check PostgreSQL status
sudo systemctl status postgresql

# Check logs
sudo tail -f /var/lib/pgsql/data/log/postgresql-*.log
```

2. **Permission Issues**

```
# Check SELinux status
sestatus

# Check file permissions
ls -la /opt/credential-grafana/captureGrafanaGIF
```

3. **Network Issues**

```
# Check ports
sudo netstat -tulpn

# Check firewall
sudo firewall-cmd --list-all
```

Backup Strategy

1. Database Backup

```
# Create backup script
vi /opt/credential-grafana/backup.sh

#!/bin/bash
BACKUP_DIR="/opt/credential-grafana/backups"
DATE=$(date +%Y%m%d_%H%M%S)
pg_dump -U postgres master_db > "$BACKUP_DIR/master_db_$DATE.sql"

# Set up daily cron job
0 0 * * * /opt/credential-grafana/backup.sh
```

2. Application Backup

```
# Backup application files
tar -czf /opt/credential-grafana/backups/app_backup_$(date +%Y%m%d).tar.gz
/opt/credential-grafana
```

Rollback Plan

1. Stop application services
2. Restore database from backup
3. Restore application files
4. Restart services

Maintenance

Regular Tasks

1. Log rotation
2. Database vacuuming
3. Backup verification
4. System updates