# A black background with a black square AI-generated content may be incorrect.

Technical Documentation

Confidential | authorized HPE partner use only

Green Matrix Deployment Guide

# Important notice

Along with Brand compliance for text and content formatting, this template has been created to ensure the correct usage of Microsoft Word functionality and meet accessibility requirements.

As such, it is **most important** that you follow the comprehensive navigation instructions for the correct method of applying the styles. Please read and use these instructions when creating a document. Correct usage also helps insure optimal conversion to PDF.

Failure to follow these instructions will result in Brand and inclusion/accessibility non-compliance.

Contents

[Executive summary 4](#_Toc207035266)

[System requirements 4](#_Toc207035267)

[Deployment options 7](#_Toc207035268)

[Docker-based deployment 8](#_Toc207035269)

[Manual installation 13](#_Toc207035270)

[VM agent deployment 17](#_Toc207035271)

[Configuration and customization 20](#_Toc207035273)

[System verification 22](#_Toc207035274)

[Troubleshooting 24](#_Toc207035275)

Security configuration……………………………………………………………………………………………………………….30

Maintenance and updates…………………………………………………………………………………………………………..32

[Please note 35](#_Toc207035296)

[External legal statement – HPE 36](#_Toc207035297)

[Internal legal statement 37](#_Toc207035298)

[Internal and channel partner legal statement 37](#_Toc207035299)

# Executive summary

GreenMatrix is an AI-powered monitoring and optimization platform designed to provide real-time insights into system performance, resource utilization, and cost optimization opportunities. This deployment guide provides comprehensive instructions for installing and configuring GreenMatrix across various infrastructure environments.

The platform supports deployment on Linux and Windows systems with provisions for both host-level and virtual machine monitoring capabilities. The platform architecture consists of multiple components including a React-based frontend, FastAPI backend services, PostgreSQL and TimescaleDB databases, and distributed monitoring agents.

## Deployment options

### Deployment Options

This guide covers three primary deployment approaches:

* **Automated Docker Deployment:** One-command installation using setup scripts
* **Manual Docker Deployment:** Step-by-step containerized deployment
* **Manual Installation:** Traditional component installation for maximum control

### Estimated Deployment Time

* **Automated Docker Deployment:** 15-20 minutes
* **Manual Docker Deployment:** 20-30 minutes
* **Manual Installation:** 45-60 minutes
* **Configuration and Testing:** Additional 15-20 minutes

# System Requirements

## Minimum Hardware Requirements

* **CPU:** 2 cores, 2.0 GHz processor
* **Memory:** 4 GB RAM
* **Storage:** 20 GB available disk space (SSD recommended)
* **Network:** 10 Mbps internet connection
* **Operating System:** Linux (Ubuntu 18.04+), Windows 10+

## Recommended Hardware Requirements

* **CPU:** 4+ cores, 2.5+ GHz processor
* **Memory:** 16+ GB RAM
* **Storage:** 50+ GB SSD storage with high IOPS capability
* **Network:** 100+ Mbps internet connection with low latency
* **Operating System:** Ubuntu 22.04 LTS, Windows Server 10+

## Production Hardware Specifications

* **CPU:** 8+ cores, 3.0+ GHz processor
* **Memory:** 32+ GB RAM
* **Storage:** 100+ GB NVMe SSD storage
* **Network:** 1+ Gbps connection
* **Operating System:** Ubuntu 22.04 LTS, Windows 10+

## Software Requirements

Prerequisites –

* **Docker:** Version 20.10+ (for Docker deployment)
* **Docker Compose:** Version 2.0+ (for Docker deployment)
* **Python:** Version 3.8+ (for manual installation)
* **Node.js:** Version 16+ (for manual installation)
* **Git:** Version 2.0+ (for code repository access)

Operating System Support

* **Linux:** Ubuntu 18.04+, CentOS 7+, RHEL 7+, Debian 10+
* **Windows:** Windows 10+, Windows Server 2016+

## Network Requirements

Port configuration -

The following network ports must be accessible:

|  |  |  |  |
| --- | --- | --- | --- |
| Service | Port | Protocol | Description |
| Web Dashboard | 80, 443 | HTTP/HTTPS | Frontend application access |
| Backend API | 8000 | HTTP | REST API services |
| PostgreSQL | 5432 | TCP | Primary database connection |
| TimescaleDB | 5433 | TCP | Time-series database connection |
| Redis | 6379 | TCP | Cache and session storage |
| Airflow | 8080 | HTTP | Workflow monitoring interface |

## Firewall Configuration-

Configure firewall rules to allow inbound connections on required ports:

### Ubuntu/Debian:

```bash

sudo ufw allow 80/tcp

sudo ufw allow 443/tcp

sudo ufw allow 8000/tcp

sudo ufw allow 5432/tcp

sudo ufw allow 5433/tcp

sudo ufw allow 6379/tcp

sudo ufw allow 8080/tcp

```

### CentOS/RHEL:

```bash

sudo firewall-cmd --permanent --add-service=http

sudo firewall-cmd --permanent --add-service=https

sudo firewall-cmd --permanent --add-port=8000/tcp

sudo firewall-cmd --permanent --add-port=5432/tcp

sudo firewall-cmd --permanent --add-port=5433/tcp

sudo firewall-cmd --permanent --add-port=6379/tcp

sudo firewall-cmd --permanent --add-port=8080/tcp

sudo firewall-cmd --reload

```

# **Deployment options:**

## Option 1: Automated Docker deployment (recommended)

- Best for: Most users, quick setup, consistent environment

- Time: 15-20 minutes

- Complexity: Low

- Maintenance: Easy

- Method: Single script execution

## Option 2: Manual Docker deployment

- Best for: Custom deployments, specific requirements

- Time: 20-30 minutes

- Complexity: Medium

- Maintenance: Moderate

- Method: Step-by-step docker-compose execution

## Option 3: Manual installation

- Best for: Maximum control and customization

- Time: 45-60 minutes

- Complexity: High

- Maintenance: High

- Method: Component-by-component installation

# **Docker-based deployment**

## Option 1: Automated Docker deployment

### Linux systems

```bash

**Clone the repository**

git clone https://github.com/your-org/greenmatrix.git

cd greenmatrix

**Run automated setup script**

chmod +x setup-greenmatrix.sh

./setup-greenmatrix.sh

```

### Windows systems

```cmd

**Clone the repository**

git clone https://github.com/your-org/greenmatrix.git

cd greenmatrix

**Run automated setup script**

setup-greenmatrix.bat

```

### What the automated script performs:

- Prerequisites verification (Docker, system resources)

- Environment configuration setup

- Directory structure creation

- Docker container building and startup

- Database initialization and population

- Host metrics collection service setup

- System health verification

## Option 2: Manual Docker deployment

### Step 1: Environment setup

```bash

**Clone the repository**

git clone https://github.com/your-org/greenmatrix.git

cd greenmatrix

**Create environment file**

cp .env.example .env

**Edit environment variables (optional)**

nano .env

```

**Key environment variables:**

```bash

**Database Configuration**

POSTGRES\_USER=postgres

POSTGRES\_PASSWORD=your\_secure\_password

POSTGRES\_PORT=5432

**Backend Configuration**

BACKEND\_PORT=8000

FRONTEND\_PORT=3000

**Airflow Configuration**

AIRFLOW\_UID=50000

```

### Step 2: Start core services

```bash

**Start database and core services**

docker-compose up -d postgres redis timescaledb

**Wait for database to be ready**

docker-compose exec -T postgres pg\_isready -U postgres

```

### Step 3: Start Airflow services

```bash

**Start Airflow database**

docker-compose up -d airflow-postgres

**Wait for Airflow database**

docker-compose exec -T airflow-postgres pg\_isready -U airflow

**Initialize Airflow**

docker-compose up airflow-init

**Start all services**

docker-compose up -d

```

### Step 4: Setup host metrics collection

```bash

**Linux Systems**

chmod +x setup-host-metrics.sh

./setup-host-metrics.sh

**Windows Systems**

setup-host-metrics.bat

```

### Step 5: Verify deployment

```bash

**Check service status**

docker-compose ps

**Test API health**

curl http://localhost:8000/health

**View logs**

docker-compose logs -f

```

### Expected service status:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Command | State | Ports |
| greenmatrix-airflow-postgres | docker-entrypoint.sh postgres | Up | 0.0.0.0:5432->5432/tcp |
| greenmatrix-backend | uvicorn views.model ... | Up | 0.0.0.0:8000->8000/tcp |
| greenmatrix-frontend | nginx -g daemon off; | Up | 0.0.0.0:8000->8000/tcp |
| greenmatrix-postgres | docker-entrypoint.sh postgres | Up | 0.0.0.0:8000->8000/tcp |
| greenmatrix-redis | docker-entrypoint.sh redis ... | Up | 0.0:6379->6379/tcp |
| greenmatrix-timescaledb | docker-entrypoint.sh postgres | Up | 0.0.0.0:5433->5432/tcp |

### Access information

|  |  |  |
| --- | --- | --- |
| Service | URL | Default Credentials |
| Web Dashboard | http://localhost:3000 | admin/admin |
| API Documentation | http://localhost:8000/docs | N/A |
| Airflow Monitoring | http://localhost:8080 | airflow/airflow |
| Database Admin | localhost:5432 | postgres/your\_password |

# **Manual installation**

## Step 1: System preparation

```bash

**Create application directory**

sudo mkdir -p /opt/greenmatrix

sudo chown $USER:$USER /opt/greenmatrix

cd /opt/greenmatrix

**Clone repository**

git clone https://github.com/your-org/greenmatrix.git .

```

## Step 2: Python environment setup

```bash

**Install Python dependencies**

sudo apt install -y python3 python3-pip python3-venv python3-dev

**Create virtual environment**

python3 -m venv venv

source venv/bin/activate

**Install Python packages**

pip install -r requirements.txt

pip install -r backend/requirements.txt

```

## Step 3: Database setup

```bash

**Install PostgreSQL**

sudo apt install -y postgresql postgresql-contrib

**Create databases**

sudo -u postgres createdb greenmatrix

sudo -u postgres createdb Model\_Recommendation\_DB

sudo -u postgres createdb Metrics\_db

**Install TimescaleDB**

sudo apt install -y timescaledb-postgresql-15

**Initialize database schema**

python -c "from backend.app.database import init\_db; init\_db()"

```

## Step 4: Backend service configuration

Create systemd service:

```bash

sudo tee /etc/systemd/system/greenmatrix-backend.service > /dev/null <<EOF

[Unit]

Description=GreenMatrix Backend API

After=network.target postgresql.service

[Service]

Type=simple

User=$USER

WorkingDirectory=/opt/greenmatrix/backend

Environment=PATH=/opt/greenmatrix/venv/bin

ExecStart=/opt/greenmatrix/venv/bin/python run.py

Restart=always

[Install]

WantedBy=multi-user.target

EOF

```

Start backend service:

```bash

sudo systemctl daemon-reload

sudo systemctl enable greenmatrix-backend

sudo systemctl start greenmatrix-backend

```

## Step 5: Frontend setup

Install Node.js:

```bash

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt install -y nodejs

```

Build frontend:

```bash

cd vite-project

npm install

npm run build

```

Install and configure Nginx:

```bash

sudo apt install -y nginx

# Configure Nginx

sudo tee /etc/nginx/sites-available/greenmatrix > /dev/null <<EOF

server {

listen 80;

server\_name localhost;

location / {

root /opt/greenmatrix/vite-project/dist;

try\_files \$uri \$uri/ /index.html;

}

location /api {

proxy\_pass http://localhost:8000;

proxy\_set\_header Host \$host;

proxy\_set\_header X-Real-IP \$remote\_addr;

}

}

EOF

**Enable site**

sudo ln -s /etc/nginx/sites-available/greenmatrix /etc/nginx/sites-enabled/

sudo nginx -t

sudo systemctl restart nginx

```

# **VM agent deployment**

**Overview**

The VM Agent is a lightweight monitoring component that runs inside virtual machines to collect performance metrics and send them to the GreenMatrix backend.

## Deployment methods

### Method 1: Automated script deployment

#### **Linux VMs:**

```bash

**Download and run deployment script**

curl -fsSL https://raw.githubusercontent.com/your-org/greenmatrix/main/deploy-vm-agent.sh | bash

**Or download first, then run**

wget https://raw.githubusercontent.com/your-org/greenmatrix/main/deploy-vm-agent.sh

chmod +x deploy-vm-agent.sh

./deploy-vm-agent.sh

```

#### **Windows VMs**

```cmd

# Download Windows deployment script

powershell -Command "Invoke-WebRequest -Uri 'https://raw.githubusercontent.com/your-org/greenmatrix/main/deploy-vm-agent.bat' -OutFile 'deploy-vm-agent.bat'"

# Run the script

deploy-vm-agent.bat

```

### Method 2: Manual installation

```bash

**Download agent files**

wget https://github.com/your-org/greenmatrix/releases/latest/download/vm-agent-package.tar.gz

tar -xzf vm-agent-package.tar.gz

cd vm-agent-package

**Install dependencies**

pip3 install -r requirements.txt

**Configure agent**

cp vm\_agent.ini.example vm\_agent.ini

nano vm\_agent.ini

**Start agent**

python3 vm\_agent.py

```

## VM agent configuration

**Key configuration options:**

```ini

[agent]

**Backend URL (GreenMatrix server)**

backend\_url = http://your-greenmatrix-server:8000

**VM identifier**

vm\_name = my-vm-01

**Collection interval (seconds)**

collection\_interval = 60

**API timeout (seconds)**

api\_timeout = 30

**CPU TDP for power estimation (watts)**

cpu\_tdp\_watts = 125.0

```

## VM agent verification

# **Configuration and customization**

**Environment configuration**

## Backend configuration

```bash

**Edit backend configuration**

nano backend/.env

**Key settings**

DATABASE\_URL=postgresql://user:password@localhost:5432/greenmatrix

MODEL\_DB\_URL=postgresql://user:password@localhost:5432/Model\_Recommendation\_DB

METRICS\_DB\_URL=postgresql://user:password@localhost:5432/Metrics\_db

TIMESCALEDB\_URL=postgresql://user:password@localhost:5433/vm\_metrics\_ts

REDIS\_URL=redis://localhost:6379

```

## Frontend configuration

```bash

**Edit frontend configuration**

nano vite-project/.env

**Key settings**

VITE\_API\_BASE\_URL=http://localhost:8000

VITE\_APP\_TITLE=GreenMatrix

VITE\_ENABLE\_DARK\_MODE=true

```

## Monitoring configuration

### Collection intervals

```bash

**Edit monitoring configuration**

nano config.ini

**Collection intervals**

[monitoring]

host\_metrics\_interval = 60 # Host metrics every 60 seconds

vm\_metrics\_interval = 30 # VM metrics every 30 seconds

process\_metrics\_interval = 10 # Process metrics every 10 seconds

```

### Alert thresholds

```bash

**CPU usage alerts**

cpu\_warning\_threshold = 80 # Warning at 80% CPU

cpu\_critical\_threshold = 95 # Critical at 95% CPU

**Memory usage alerts**

memory\_warning\_threshold = 85 # Warning at 85% memory

memory\_critical\_threshold = 95 # Critical at 95% memory

```

# **System verification**

**Health check commands**

## Service status verification

```bash

**Check all services**

docker-compose ps

**Check specific service**

docker-compose logs greenmatrix-backend

**Check service health**

curl -f http://localhost:8000/health

```

## Database connectivity

```bash

**Test PostgreSQL connection**

psql -h localhost -U postgres -d greenmatrix -c "SELECT version();"

**Test TimescaleDB connection**

psql -h localhost -U postgres -d vm\_metrics\_ts -c "SELECT version();"

**Check database sizes**

psql -h localhost -U postgres -c "\l+"

```

## API endpoint testing

```bash

**Test main API**

curl http://localhost:8000/api/status

**Test metrics endpoint**

curl http://localhost:8000/api/monitoring/metrics

**Test hardware endpoint**

curl http://localhost:8000/api/hardware/

```

## Performance testing

### Load testing

```bash

**Install testing tool**

pip install locust

**Run load test**

locust -f load\_test.py --host=http://localhost:8000

```

### Benchmark testing

```bash

**Run performance benchmarks**

python scripts/performance\_benchmark.py

**Check response times**

curl -w "@curl-format.txt" -o /dev/null -s "http://localhost:8000/health"

```

# **Troubleshooting**

## Common issues and solutions

### Issue 1: Services won't start

Symptoms:

- Docker containers exit immediately

- Systemd services fail to start

- Port conflicts

Solutions:

```bash

**Check container logs**

docker-compose logs [service-name]

**Check system resources**

free -h

df -h

nproc

**Check port conflicts**

sudo netstat -tulpn | grep :8000

**Restart Docker daemon**

sudo systemctl restart docker

```

### Issue 2: Database connection errors

Symptoms:

- "Connection refused" errors

- Authentication failures

- Database not found errors

Solutions:

```bash

**Check PostgreSQL status**

sudo systemctl status postgresql

**Check database existence**

sudo -u postgres psql -l

**Verify connection parameters**

cat .env | grep DATABASE

**Test manual connection**

psql -h localhost -U postgres -d greenmatrix

```

### Issue 3: Frontend not loading

Symptoms:

- Blank page

- JavaScript errors

- API connection failures

Solutions:

```bash

**Check frontend build**

ls -la vite-project/dist/

**Check Nginx configuration**

sudo nginx -t

**Check frontend logs**

docker-compose logs greenmatrix-frontend

**Verify API connectivity**

curl http://localhost:8000/health

```

### Issue 4: VM agent issues

Symptoms:

- No VM metrics in dashboard

- Connection timeouts

- Authentication errors

Solutions:

```bash

**Check agent status**

sudo systemctl status greenmatrix-vm-agent

**Check agent logs**

sudo journalctl -u greenmatrix-vm-agent -f

**Test network connectivity**

ping your-greenmatrix-server

**Verify backend URL**

cat /opt/greenmatrix-vm-agent/vm\_agent.ini

```

## Log analysis

### Understanding logs

```bash

**Backend logs**

docker-compose logs -f greenmatrix-backend

**Frontend logs**

docker-compose logs -f greenmatrix-frontend

**Database logs**

docker-compose logs -f greenmatrix-postgres

**Airflow logs**

docker-compose logs -f greenmatrix-airflow-webserver

```

### Log levels

```bash

**Set log level in .env**

LOG\_LEVEL=DEBUG # DEBUG, INFO, WARNING, ERROR

**Filter logs by level**

docker-compose logs greenmatrix-backend | grep "ERROR"

```

## Performance issues -

### High CPU usage

```bash

**Check container resource usage**

docker stats

**Identify resource-intensive processes**

docker-compose exec backend top

**Optimize database queries**

docker-compose exec postgres psql -U postgres -d greenmatrix -c "SELECT \* FROM pg\_stat\_activity;"

```

### Memory issues -

```bash

**Check memory usage**

free -h

**Check container memory limits**

docker-compose exec backend cat /proc/meminfo

```

### Network issues -

```bash

**Check network connectivity**

ping google.com

**Test port accessibility**

telnet localhost 8000

**Check firewall rules**

sudo ufw status

```

# **Security configuration**

## Basic security setup

## Firewall configuration

```bash

**Ubuntu/Debian**

sudo ufw enable

sudo ufw allow ssh

sudo ufw allow 80/tcp

sudo ufw allow 443/tcp

sudo ufw allow 8000/tcp

**CentOS/RHEL**

sudo firewall-cmd --permanent --add-service=ssh

sudo firewall-cmd --permanent --add-service=http

sudo firewall-cmd --permanent --add-service=https

sudo firewall-cmd --permanent --add-port=8000/tcp

sudo firewall-cmd --reload

```

**SSL/TLS configuration**

```bash

**Install Certbot**

sudo apt install -y certbot python3-certbot-nginx

**Obtain SSL certificate**

sudo certbot --nginx -d your-domain.com

**Auto-renewal**

sudo crontab -e

# Add: 0 12 \* \* \* /usr/bin/certbot renew --quiet

```

## Advanced security

## Database security

```bash

**Change default passwords**

sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'new\_secure\_password';"

**Restrict database access**

sudo nano /etc/postgresql/15/main/pg\_hba.conf

**Enable SSL**

sudo nano /etc/postgresql/15/main/postgresql.conf

```

## API security

Enable API authentication by editing backend/views/model\_api\_routes.py and adding rate limiting capabilities.

> **Note**

> For production deployments, ensure all default passwords are changed and proper authentication mechanisms are implemented.

# **Maintenance and updates**

## Regular maintenance tasks

### Daily tasks

```bash

**Check service status**

docker-compose ps

**Check disk usage**

df -h

**Check log sizes**

du -sh logs/\*/

```

### Weekly tasks

```bash

**Database maintenance**

docker-compose exec postgres psql -U postgres -c "VACUUM ANALYZE;"

**Log rotation**

sudo logrotate /etc/logrotate.d/greenmatrix

**Backup verification**

./scripts/verify\_backup.sh

```

### Monthly tasks

```bash

**System updates**

sudo apt update && sudo apt upgrade -y

**Docker image updates**

docker-compose pull

docker-compose up -d

**Performance analysis**

python scripts/performance\_analysis.py

```

## Backup and recovery

## Database backup

Create backup script:

```bash

cat > backup\_db.sh << 'EOF'

#!/bin/bash

BACKUP\_DIR="/opt/backups/greenmatrix"

DATE=$(date +%Y%m%d\_%H%M%S)

mkdir -p $BACKUP\_DIR

**Backup PostgreSQL databases**

docker-compose exec -T postgres pg\_dump -U postgres greenmatrix > $BACKUP\_DIR/greenmatrix\_$DATE.sql

docker-compose exec -T postgres pg\_dump -U postgres Model\_Recommendation\_DB > $BACKUP\_DIR/model\_db\_$DATE.sql

docker-compose exec -T postgres pg\_dump -U postgres Metrics\_db > $BACKUP\_DIR/metrics\_db\_$DATE.sql

**Backup TimescaleDB**

docker-compose exec -T timescaledb pg\_dump -U postgres vm\_metrics\_ts > $BACKUP\_DIR/timescaledb\_$DATE.sql

echo "Backup completed: $BACKUP\_DIR"

EOF

chmod +x backup\_db.sh

```

## System recovery

```bash

**Restore from backup**

docker-compose exec -T postgres psql -U postgres greenmatrix < backup\_file.sql

**Restart services**

docker-compose restart

**Verify restoration**

curl http://localhost:8000/health

```

## Update procedures

### Application updates

```bash

**Pull latest code**

git pull origin main

**Rebuild containers**

docker-compose build --no-cache

**Restart services**

docker-compose down

docker-compose up -d

R**un database migrations**

docker-compose exec backend python -c "from app.database import migrate\_db; migrate\_db()"

```

### Dependency updates

```bash

**Update Python dependencies**

pip install -r requirements.txt --upgrade

**Update Node.js dependencies**

cd vite-project

npm update

**Rebuild frontend**

npm run build

```Web Dashboard

# Please note

Check if there are any third-party trademarks referenced in the document that require specific attribution under contractual agreements (certain third parties require their individual marks to be listed). If so, the individual trademarks and their owners should be listed instead of the yellow highlighted language, and then the yellow highlighted portion of the notice should be deleted before publication. The generic statement highlighted in green at the end should always remain, even if there are no specific contractual requirements to list specific marks: Trademark acknowledgments, if needed. All third-party marks are property of their respective owners.

# External legal statement – HPE

Use the below for documents with the HPE logo in the header.

|  |  |
| --- | --- |
|  | [A close-up of a logo  AI-generated content may be incorrect.](http://hpe.com/) |
|  | [Chat now](https://www.hpe.com/global/hpechat/index.html?jumpid=Collaterals_aXXXXXXXXENW)  © Copyright 2025 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.  Trademark acknowledgments, if needed. All third-party marks are property of their respective owners.  aXXXXXXXXENW  HPE green elementHEWLETT PACKARD ENTERPRISE  Hpe.com |

# Internal legal statement

|  |  |
| --- | --- |
|  | © Copyright 2025 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.  This document contains confidential and/or legally privileged information. It is intended for Hewlett Packard Enterprise Internal Use only. If you are not an intended recipient as identified on the front cover of this document, you are strictly prohibited from reviewing, redistributing, disseminating, or in any other way using or relying on the contents of this document.  Trademark acknowledgments, if needed. All third-party marks are property of their respective owners.  aXXXXXXXXENW  HPE green elementHEWLETT PACKARD ENTERPRISE  Hpe.com |

# Internal and channel partner legal statement

|  |  |
| --- | --- |
|  | © Copyright 2025 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.  This document contains confidential and/or legally privileged information. It is intended for Hewlett Packard Enterprise and Channel Partner Internal Use only. If you are not an intended recipient as identified on the front cover of this document, you are strictly prohibited from reviewing, redistributing, disseminating, or in any other way using or relying on the contents of this document.  Trademark acknowledgments, if needed. All third-party marks are property of their respective owners.  aXXXXXXXXENW  HPE green elementHEWLETT PACKARD ENTERPRISE  Hpe.com |