E(i)xM installation requirements

Version **1.0**

Date **12/02/2026**

GENERAL INFORMATION 2

How many containers are defined in the docker compose config? 2

Database: product, version, expected size 2

Storage separation - application vs database 2

LV size for /opt/eixm? 3

Separate LV for /opt/eixm/storage and size? 3

Possibility for upfront delivery of docker compose config? 4

TECHNICAL MD DOCUMENT 5

GENERAL INFORMATION

How many containers are defined in the docker compose config?

15 containers total:

Core Services (5):

- eixm-api - Main API application
- eixm-api-legacy - Legacy API for backward compatibility
- eixm-sapmoni-api - SAP MONI integration API
- eixm-proxy - Reverse proxy
- eixm-web - Web frontend

Infrastructure (10):

- reverse-proxy - Traefik reverse proxy (HTTPS/SSL termination)
- docker-socket-proxy - Security proxy for Docker socket
- mssql-server - Microsoft SQL Server database
- mssql-init - Database initialization (one-time)
- mssql-backup - Automated database backups
- redis - Cache and session storage
- seq - Structured logging server
- env-dashboard - Dashboard for monitoring
- eixm-db-migrations - Database schema migrations (one-time)
- app-init - Application initialization (one-time)

Database: product, version, expected size

Product: Microsoft SQL Server
Version: 2022-latest
Image: mcr.microsoft.com/mssql/server:2022-latest

Expected database size:

- Initial: ~500 MB - 1 GB
- Growth: Depends on usage
- Recommended: Plan for 50-100 GB minimum for production use
- Backups: Full + transaction log backups (configure retention period)

Part of docker compose: Yes, deployed as mssql-server container

Storage separation - application vs database

No, storage is separated:

Database storage:

- Docker volume: eixm-mssql-data (database files)
- Docker volume: eixm-mssql-backups (database backups)
- NOT stored in /opt/eixm/storage

Application file storage:

- Docker volume: eixm-storage mounted to /opt/eixm/storage
- Stores application related data (for example: uploaded documents, files, attachments)

Other storage:

- eixm-seq-data - Log data
- eixm-redis-data - Cache data
- letsencrypt - SSL certificates

LV size for /opt/eixm?

Recommended: 20-25 GB (for air-gapped deployment)

Contents:

Permanent files (~100 KB):

- Docker Compose files (~50 KB)
- Deployment scripts (~30 KB)
- Configuration files (.env, etc.) (~5 KB)
- SSL certificates (~10 KB)
- Deployment artifacts (~100 KB)

Temporary files during air-gapped deployment (~10-15 GB):

- Air-gapped bundle archive: ~5-10 GB compressed
- Extracted bundle contents: ~5-10 GB (images as tar.gz files)
- Working space for extraction: ~2-3 GB

Note: After initial deployment, you can reduce to 5 GB if needed, but keep 20-25 GB to accommodate future updates without resizing.

Separate LV for /opt/eixm/storage and size?

Not required - /opt/eixm/storage is just a mount point for Docker volume eixm-storage.

Recommendation: Create separate LV for Docker volumes instead:

Option 1: Separate LV for all Docker volumes (recommended)

- Mount point: /var/lib/docker/volumes
- Size: 150-200 GB minimum
- Contains all Docker volumes (database, backups, storage, logs, cache)

Option 2: Separate LVs per volume type

- /var/lib/docker/volumes/eixm-mssql-data - 100 GB (database)
- /var/lib/docker/volumes/eixm-mssql-backups - 50 GB (backups)
- /var/lib/docker/volumes/eixm-storage - 50 GB (application files)
- /var/lib/docker/volumes/eixm-seq-data - 20 GB (logs)
- Others - 10 GB total

Total recommended: 200-250 GB for all Docker volumes, in case ERP is not used 25 GB are ok

Possibility for upfront delivery of docker compose config?

Yes, we can provide the following for gap analysis:

- docker-compose.yaml - Main configuration
- docker-compose.airgap.yaml - Air-gapped deployment overlay
- .env - Environment variables template

Additional files available:

- backup-db.sh - Backup automation script
- airgap-import.sh - Docker image import script
- deploy-airgap.sh - Air-gapped deployment script
- generate-self-signed-cert.sh - Certificate generation script
- import-custom-cert.sh - Custom certificate import script

TECHNICAL MD DOCUMENT

EIXM System Preparation Guide

Overview

This document describes how to prepare your Linux VM for EIXM deployment. Follow these instructions to set up the required user account, permissions, and directory structure before deploying EIXM.

Supported Linux Distributions:
- SUSE Linux Enterprise Server (SLES) 15+
- openSUSE Leap 15+
- Ubuntu 22.04+
- Red Hat Enterprise Linux (RHEL) 8+
- Debian 11+
- Other systemd-based distributions with Docker support

Note: SUSE-specific configuration notes are included throughout this document where applicable.

System Requirements

Your Linux VM must have:
- **OS:** Linux distribution with systemd (see supported list above)
- **Docker:** Docker Engine 20.10+ or Docker CE 20.10+
- **Docker Compose:** Docker Compose V2 (plugin or standalone)
- **RAM:** 16+ GB
- **CPU:** 4+ cores
- **Storage:** 100+ GB available disk space
- **Time Sync:** NTP/chrony configured (for accurate logs and scheduled jobs)
- **Network:** Outbound access to container registries (or air-gapped deployment option)

Deployment User Setup

EIXM requires a dedicated non-root user account for deployment and operation.

Create the Deployment User

Recommended username: `deploy` (or `eixmdeploy`)

```bash
# Run as root or with sudo
sudo useradd -m -s /bin/bash deploy
sudo passwd deploy  # Set a strong password
```

Note: You can use any username you prefer. Replace `deploy` with your chosen username throughout this guide.

SSH Access Configuration

The deployment user requires SSH access with public key authentication.

EIXM vendor will provide you with a public SSH key to add to your deployment user's authorized keys.

Configure SSH access on the VM:

```bash
# Switch to deployment user (on the VM)
sudo su - deploy

# Create .ssh directory
mkdir -p ~/.ssh
chmod 700 ~/.ssh

# Add the public SSH key provided by EIXM vendor
# Replace <public-key-from-vendor> with the actual key provided
echo "<public-key-from-vendor>" >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
exit
```

Security recommendations:
- ☑ **Use public key authentication** (no password login)
- ☑ **Only add public keys** from trusted sources (your EIXM vendor)
- ☑ **Disable password authentication** for SSH (optional but recommended)
- ❌ **Do NOT enable root SSH access**

To disable password authentication (optional but recommended):

```bash
# Edit SSH config (run as root)
sudo nano /etc/ssh/sshd_config

# Set these values:
# PasswordAuthentication no
# PubkeyAuthentication yes
# PermitRootLogin no

# Restart SSH service
sudo systemctl restart sshd  # or 'ssh' on some distributions
```

After configuration, provide these details to your EIXM vendor:
- VM hostname or IP address
- SSH port (default: 22)
- Deployment username (e.g., `deploy`)
- Any firewall rules restricting source IPs

Required Permissions

1. Docker Access (Critical)

The deployment user must be able to execute Docker commands without `sudo`.

Setup:
```bash
# Add user to docker group (run as root)
sudo usermod -aG docker deploy

# User must log out and log back in for group membership to take effect
# Then verify docker access
docker ps
docker --version
docker compose version
```

Required Docker capabilities:

The deployment user needs access to the following Docker commands:
- `docker ps` - List running containers
- `docker pull` - Pull images from registries
- `docker run` - Run containers
- `docker compose up/down` - Manage multi-container applications
- `docker logs` - View container logs
- `docker exec` - Execute commands in running containers
- `docker images` - List images
- `docker load` - Load images from tar archives (air-gapped deployments)

Verify Docker access:

```bash
# Run these commands as the deployment user
docker ps
docker images
docker compose version
docker network ls
```

Alternative: Minimal sudo permissions (if Docker group access is restricted)

If your security policy does not allow adding the user to the `docker` group, configure minimal sudo permissions:

```bash
# Create sudoers file for deployment user (run as root)
sudo visudo -f /etc/sudoers.d/deploy

# Add these lines (replace 'deploy' with your username):
deploy ALL=(ALL) NOPASSWD: /usr/bin/docker
deploy ALL=(ALL) NOPASSWD: /usr/bin/docker-compose
deploy ALL=(ALL) NOPASSWD: /usr/local/bin/docker-compose
```

```
deploy ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart docker
deploy ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop docker
deploy ALL=(ALL) NOPASSWD: /usr/bin/systemctl start docker

# Save and exit
# Set proper permissions
sudo chmod 440 /etc/sudoers.d/deploy
```

Note: Docker group membership is the **preferred and recommended** approach. The sudo alternative should only be used if required by your organization's security policies.

2. File System Access

EIXM requires specific directories for storing configuration, application data, and backups.

Create directory structure:

```bash
# Create directories (run as root)
sudo mkdir -p /opt/eixm
sudo mkdir -p /opt/eixm/certs
sudo mkdir -p /opt/eixm/storage
sudo mkdir -p /opt/eixm/backups

# Set ownership to deployment user (replace 'deploy' with your username)
sudo chown -R deploy:deploy /opt/eixm

# Set permissions
sudo chmod 755 /opt/eixm
sudo chmod 700 /opt/eixm/certs  # Certificates are sensitive
sudo chmod 775 /opt/eixm/storage
sudo chmod 775 /opt/eixm/backups
```

Directory purposes:
- `/opt/eixm/` - Deployment files and configuration
- `/opt/eixm/certs/` - SSL certificates and private keys
- `/opt/eixm/storage/` - Application file storage (Docker volume mount)
- `/opt/eixm/backups/` - Database backups

Permissions breakdown:

Path	Owner	Group	Permissions	Purpose
`/opt/eixm/`	deploy	deploy	755 (rwxr-xr-x)	Deployment files
`/opt/eixm/certs/`	deploy	deploy	700 (rwx------)	SSL certificates
`/opt/eixm/storage/`	deploy	deploy	775 (rwxrwxr-x)	Application storage
`/opt/eixm/backups/`	deploy	deploy	775 (rwxrwxr-x)	Database backups
`/opt/eixm/.env`	deploy	deploy	600 (rw-------)	Configuration secrets

3. Network Configuration

Required Network Access:

Inbound:
- **SSH (port 22)** - For deployment access (can be restricted by source IP if required)
- **HTTPS (port 443)** - For users to access the EIXM application
- **HTTP (port 80)** - Optional, for HTTP to HTTPS redirect

Outbound:
- **Docker image registries** - For pulling container images (HTTPS/443)

Note: For air-gapped deployments, outbound access is not required.

Firewall Setup:

For RHEL/CentOS/SUSE (firewalld):

```bash
# Allow HTTPS traffic (run as root)
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --permanent --add-service=http  # Optional
sudo firewall-cmd --reload

# Verify firewall configuration
sudo firewall-cmd --list-all
```

For Ubuntu/Debian (ufw):

```bash
# Allow HTTPS traffic (run as root)
sudo ufw allow 443/tcp
sudo ufw allow 80/tcp  # Optional
sudo ufw reload

# Verify firewall status
sudo ufw status
```

For SUSE (SuSEfirewall2 on older versions):

```bash
# For SLES 12 and older
sudo SuSEfirewall2 open EXT TCP 443
sudo SuSEfirewall2 open EXT TCP 80  # Optional
sudo SuSEfirewall2
```

Restrict SSH by source IP (optional but recommended):

```bash
# Example: Allow SSH only from specific IP
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="YOUR.DEPLOY.IP.ADDRESS" service name="ssh" accept'
```

sudo firewall-cmd --reload
```

### 4. OpenSSL Access

**Required for certificate operations:**

```bash
# Verify OpenSSL is available
openssl version
```

The deployment user must be able to execute OpenSSL commands for:
- Generating self-signed certificates (if not using organization CA)
- Verifying certificate and key pairs
- Inspecting certificate details

5. Time Synchronization

Critical for logs, scheduled jobs, and security:

```bash
# Check time synchronization status
timedatectl

# Verify NTP/chrony is active
systemctl status chronyd  # RHEL/CentOS/SUSE
# or
systemctl status systemd-timesyncd  # Ubuntu/Debian
```

If time sync is not configured:

```bash
# For RHEL/CentOS/SUSE (chrony)
sudo yum install chrony  # or zypper install chrony
sudo systemctl enable --now chronyd

# For Ubuntu/Debian (systemd-timesyncd)
sudo systemctl enable --now systemd-timesyncd

# Verify time is synced
timedatectl status
```

Recommended timezone: UTC (but any timezone is acceptable if consistent)

```bash
# Set timezone to UTC (optional)
sudo timedatectl set-timezone UTC

# Or set to your preferred timezone
sudo timedatectl set-timezone Europe/Zurich
```
```

```

### 6. Optional but Recommended Utilities

**Install helpful utilities:**

```bash
# For RHEL/CentOS/SUSE
sudo yum install curl jq ca-certificates  # or zypper install

# For Ubuntu/Debian
sudo apt-get install curl jq ca-certificates
```

These utilities are used for:
- `curl` - API health checks and troubleshooting
- `jq` - JSON parsing and log analysis
- `ca-certificates` - SSL/TLS certificate validation

What We Do NOT Require

To clarify the scope of access needed, EIXM deployment does **NOT** require:

- ❌ **Root SSH access** - A dedicated non-root user is sufficient
- ❌ **Full sudo privileges** - Only Docker and minimal system commands (if needed)
- ❌ **Manual intervention during deployments** - Deployments are automated
- ❌ **Access to other VMs or infrastructure** - Only this VM is required
- ❌ **Database root credentials** - Application uses dedicated database user
- ❌ **Modifications to system-wide configurations** - Only user-specific and application-specific settings

---

SUSE Linux Specific Configuration

AppArmor

SUSE uses AppArmor for security. If you encounter permission denied errors with Docker:

```bash
# Check AppArmor status
sudo aa-status

# If needed, put Docker in complain mode
sudo aa-complain /etc/apparmor.d/docker
```

Firewall (firewalld)

SUSE 15+ uses `firewalld`:
```

```bash
# Check firewall status
sudo firewall-cmd --state

# List current configuration
sudo firewall-cmd --list-all
```

SELinux (if enabled)

Some SUSE installations may have SELinux enabled:

```bash
# Check SELinux status
getenforce

# If enforcing and causing issues with Docker
sudo semanage permissive -a container_t
```

Verification

Automated Verification Script

Save this script as `verify-deployment-permissions.sh` and run it to verify all required permissions are in place:

```bash
#!/bin/bash

# EIXM Deployment Permission Verification Script
# Usage: ./verify-deployment-permissions.sh [airgap|azure]

MODE=${1:-azure}
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m'

echo "========================================="
echo "EIXM Deployment Permission Verification"
echo "Mode: $MODE"
echo "User: $(whoami)"
echo "========================================="
echo ""

ERRORS=0

# Check Docker access
echo "1. Checking Docker access..."
```

```
if docker ps > /dev/null 2>&1; then
   echo -e "   ${GREEN}✓${NC} Docker access: OK"
else
   echo -e "   ${RED}✗${NC} Docker access: FAILED"
   echo "      User must be in docker group: sudo usermod -aG docker $(whoami)"
   ERRORS=$((ERRORS + 1))
fi

# Check Docker Compose
echo "2. Checking Docker Compose..."
if docker compose version > /dev/null 2>&1; then
   echo -e "   ${GREEN}✓${NC} Docker Compose V2: OK"
else
   echo -e "   ${RED}✗${NC} Docker Compose: FAILED"
   ERRORS=$((ERRORS + 1))
fi

# Check OpenSSL
echo "3. Checking OpenSSL..."
if openssl version > /dev/null 2>&1; then
   echo -e "   ${GREEN}✓${NC} OpenSSL: OK ($(openssl version))"
else
   echo -e "   ${RED}✗${NC} OpenSSL: FAILED"
   ERRORS=$((ERRORS + 1))
fi

# Check /opt/eixm directory
echo "4. Checking /opt/eixm directory..."
if [ -d "/opt/eixm" ]; then
   if [ -w "/opt/eixm" ]; then
      echo -e "   ${GREEN}✓${NC} /opt/eixm: Exists and writable"
   else
      echo -e "   ${RED}✗${NC} /opt/eixm: Not writable by $(whoami)"
      ERRORS=$((ERRORS + 1))
   fi
else
   echo -e "   ${YELLOW}⚠${NC} /opt/eixm: Does not exist (will be created)"
fi

# Check storage directory
echo "5. Checking /opt/eixm/storage directory..."
if [ -d "/opt/eixm/storage" ]; then
   if [ -w "/opt/eixm/storage" ]; then
      echo -e "   ${GREEN}✓${NC} /opt/eixm/storage: Exists and writable"
   else
      echo -e "   ${RED}✗${NC} /opt/eixm/storage: Not writable"
      ERRORS=$((ERRORS + 1))
   fi
else
   echo -e "   ${YELLOW}⚠${NC} /opt/eixm/storage: Does not exist (will be created)"
fi
```

```
# Check backup directory
echo "6. Checking /opt/eixm/backups directory..."
if [ -d "/opt/eixm/backups" ]; then
    if [ -w "/opt/eixm/backups" ]; then
        echo -e "   ${GREEN}✓${NC} /opt/eixm/backups: Exists and writable"
    else
        echo -e "   ${RED}✗${NC} /opt/eixm/backups: Not writable"
        ERRORS=$((ERRORS + 1))
    fi
else
    echo -e "   ${YELLOW}⚠${NC} /opt/eixm/backups: Does not exist (will be created)"
fi

# Air-gapped specific checks
if [ "$MODE" = "airgap" ]; then
    echo "7. Checking /opt/eixm/certs directory (airgap)..."
    if [ -d "/opt/eixm/certs" ]; then
        if [ -w "/opt/eixm/certs" ]; then
            echo -e "   ${GREEN}✓${NC} /opt/eixm/certs: Exists and writable"
        else
            echo -e "   ${RED}✗${NC} /opt/eixm/certs: Not writable"
            ERRORS=$((ERRORS + 1))
        fi
    else
        echo -e "   ${YELLOW}⚠${NC} /opt/eixm/certs: Does not exist (will be created)"
    fi

    echo "8. Testing Docker image load capability..."
    if docker load --help > /dev/null 2>&1; then
        echo -e "   ${GREEN}✓${NC} Can execute 'docker load'"
    else
        echo -e "   ${RED}✗${NC} Cannot execute 'docker load'"
        ERRORS=$((ERRORS + 1))
    fi
fi

# Check disk space
echo "9. Checking disk space..."
AVAILABLE_GB=$(df -BG /opt 2>/dev/null | awk 'NR==2 {print $4}' | sed 's/G//')
if [ -z "$AVAILABLE_GB" ]; then
    AVAILABLE_GB=$(df -BG / | awk 'NR==2 {print $4}' | sed 's/G//')
fi

if [ "$AVAILABLE_GB" -ge 100 ]; then
    echo -e "   ${GREEN}✓${NC} Disk space: ${AVAILABLE_GB}GB available"
elif [ "$AVAILABLE_GB" -ge 50 ]; then
    echo -e "   ${YELLOW}⚠${NC} Disk space: ${AVAILABLE_GB}GB available (Low, recommend
100GB+)"
else
    echo -e "   ${RED}✗${NC} Disk space: ${AVAILABLE_GB}GB available (Insufficient)"
    ERRORS=$((ERRORS + 1))
```

```
fi

# Check firewall (informational)
echo "10. Checking firewall (informational)..."
if command -v firewall-cmd > /dev/null 2>&1; then
   if sudo -n firewall-cmd --state > /dev/null 2>&1; then
      if sudo firewall-cmd --list-services | grep -q https; then
         echo -e "   ${GREEN}✓${NC} Firewall: HTTPS is allowed"
      else
         echo -e "   ${YELLOW}⚠${NC} Firewall: HTTPS not explicitly allowed (may be OK)"
      fi
   else
      echo -e "   ${YELLOW}⚠${NC} Firewall: Cannot check (insufficient sudo privileges)"
   fi
else
   echo -e "   ${YELLOW}⚠${NC} Firewall: firewall-cmd not found"
fi

echo ""
echo "========================================="
if [ $ERRORS -eq 0 ]; then
   echo -e "${GREEN}✓ All checks passed!${NC}"
   echo "The user $(whoami) has the required permissions for EIXM deployment."
   exit 0
else
   echo -e "${RED}✗ $ERRORS error(s) found${NC}"
   echo "Please fix the errors above before proceeding with deployment."
   exit 1
fi
```

Usage:

```bash
chmod +x verify-deployment-permissions.sh
./verify-deployment-permissions.sh
```

Quick Setup Guide

Follow these steps to prepare your Linux VM for EIXM deployment:

```bash
# 1. Create user and add to docker group (run as root on the VM)
sudo useradd -m -s /bin/bash deploy
sudo usermod -aG docker deploy

# 2. Set up SSH access for deployment user (on the VM)
# First, obtain the public SSH key from your EIXM vendor
sudo su - deploy
```

```
mkdir -p ~/.ssh
chmod 700 ~/.ssh
echo "<public-key-from-vendor>" >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
exit

# 3. Create directory structure (run as root)
sudo mkdir -p /opt/eixm/{certs,storage,backups}
sudo chown -R deploy:deploy /opt/eixm
sudo chmod 755 /opt/eixm
sudo chmod 700 /opt/eixm/certs
sudo chmod 775 /opt/eixm/{storage,backups}

# 4. Configure firewall (RHEL/CentOS/SUSE)
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload

# For Ubuntu/Debian (ufw)
# sudo ufw allow 443/tcp
# sudo ufw reload

# 5. Verify time synchronization
timedatectl status

# 6. Install optional utilities
sudo yum install curl jq ca-certificates  # RHEL/CentOS/SUSE
# or
# sudo apt-get install curl jq ca-certificates  # Ubuntu/Debian

# 7. Verify Docker access (as deploy user)
su - deploy
docker ps
docker compose version
cd /opt/eixm
openssl version
```

Next steps:
1. ☑ Run the verification script (see Verification section above)
2. ☑ Provide SSH access details to EIXM vendor:
   - VM hostname/IP address
   - SSH port (if non-standard)
   - Deployment username
   - Any IP restrictions
3. ☑ Coordinate deployment schedule

Troubleshooting

"Permission denied" when running docker commands

Problem: User is not in docker group or hasn't logged out/in after being added.

Solution:
```bash
# Verify group membership
id
groups

# If docker group is missing, add user (as root)
sudo usermod -aG docker $(whoami)

# Log out and log back in
exit
# Then log in again
```

Alternative: Verify the Docker daemon is running:

```bash
sudo systemctl status docker
```

"Cannot connect to Docker daemon"

Problem: Docker service is not running or user doesn't have access.

Solution:
```bash
# Check Docker service status
sudo systemctl status docker

# Start Docker if not running
sudo systemctl start docker
sudo systemctl enable docker
```

"Permission denied" for /opt/eixm

Problem: Directory ownership is incorrect.

Solution:
```bash
# Fix ownership (run as root, replace 'deploy' with your username)
sudo chown -R deploy:deploy /opt/eixm
```

SSH key authentication not working

Problem: Public key is not properly configured.

Solution:
```bash
# Check .ssh directory permissions (as deployment user)
```

```
ls -la ~/.ssh
# Should show: drwx------ (700) for .ssh directory
# Should show: -rw------- (600) for authorized_keys file

# Fix permissions if needed
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys

# Verify SSH daemon allows public key auth (as root)
sudo grep "PubkeyAuthentication" /etc/ssh/sshd_config
# Should show: PubkeyAuthentication yes
```

Firewall blocking access

Problem: Cannot access application on port 443.

Solution:
```bash
# Check firewall status and rules
sudo firewall-cmd --list-all  # RHEL/CentOS/SUSE
# or
sudo ufw status  # Ubuntu/Debian

# Verify HTTPS is allowed
sudo firewall-cmd --list-services | grep https
# or
sudo ufw status | grep 443
```

Security Recommendations

1. **Dedicated User:** Use a dedicated service account for EIXM, not a personal account
2. **Firewall Rules:** Only open required ports (443 for HTTPS)
3. **Audit Logs:** Enable and monitor audit logs for Docker operations
4. **Regular Updates:** Keep the system and Docker updated with security patches
5. **Certificate Security:** Protect private keys with 600 permissions
6. **Secrets Management:** Never commit `.env` files or secrets to version control
7. **Least Privilege:** Only grant minimum required permissions

---

Contact

For questions or issues related to system preparation:
- Contact your EIXM vendor or support team

Document Version: 1.0

Last Updated: 2026-01-27
Applies To: EIXM v1.0.0+ on Linux with Docker pre-installed
Distributions: SUSE, RHEL, Ubuntu, Debian, and other systemd-based distributions