PODMAN - Install Podman on Linux Redhat

Production-Ready Step-by-Step Guide Rootless Podman + Quadlet on Red Hat Enterprise Linux 10

This guide is designed for a production environment as assums a fresh install of Redhat. It uses the modern **Quadlet** method (recommended) and follows Red Hat best practices for RHEL 10.

---

1. System Requirements (Production)

Resource	Minimum	Recommended (Production)	Our Systems
CPU	2 cores	4+ cores	8 cores
RAM	4 GB	8–16 GB+	32 GB
Disk	40 GB free	100 GB+ (SSD preferred)	350 GB
cgroup	cgroup v2	cgroup v2 (required)	
SELinux	Enforcing	Enforcing	

Check cgroup version:

```
podman info --format '{{.Host.CgroupVersion}}'
```

Install Needed Tools:

Epel Release

```
subscription-manager repos --enable codeready-builder-for-rhel-10-$(arch)-rpms
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm -y
```

After Epel installation rerun the upgrade to update if any are needed

```
dnf upgrade -y
```

Toolset

```
dnf install bind-utils bzip2 cups cifs-utils enscript ftp gdb ghostscript krb5-workstation ksh lftp lrzsz lsof libnsl lzop plocate mutt ncompress net-tools net-snmp net-snmp-utils net-tools nfs-utils nmap nvme-cli openldap-clients openssh-clients psmisc realmd rsync samba-client strace sysstat tcpdump telnet telnet-server tmux unix2dos vim vim-enhanced vsftpd wget xfsdump vsftpd htop mc rsyslog rsyslog-doc postfix dbus-daemon s-nail dovecot cyrus-sasl cyrus-sasl-lib cyrus-sasl-plain tree figlet toilet coreutils -y
```

---

2. Prepare the System

2.1 Register and Update RHEL 10

```
sudo dnf update -y
sudo reboot
```

2.2 Install Container Tools

```
sudo dnf install -y container-tools
```

Optional (Docker CLI compatibility):

```
sudo dnf install -y podman-docker
```

Verify:

```
podman --version
podman info
```

---

3. Create a Dedicated Service User (Best Practice)

For production, avoid running services under a regular interactive user.

```
# Create a system user for containers
sudo useradd -r -m -d /home/podman -s /bin/bash podman
```

```
# Set a strong password or disable password login
sudo passwd podman          # or lock the account later
```

Grant subordinate UIDs/GIDs (required for rootless):

```
sudo usermod --add-subuids 100000-165535 --add-subgids 100000-165535 podman
```

---

4. Enable Lingering (Critical for Production)

This allows the user services to run even when the user is not logged in.

```
sudo loginctl enable-linger podman
```

Verify:

```
loginctl show-user podman | grep Linger
# Linger=yes
```

---

5. Configure Rootless Environment

Switch to the service user:

```
sudo -u podman -i
```

Create required directories:

```
mkdir -p ~/.config/containers/systemd
mkdir -p ~/.config/containers
mkdir -p ~/.local/share/containers
```

Optional but Recommended: Storage Configuration

```
cat > ~/.config/containers/storage.conf << 'EOF'
[storage]
driver = "overlay"
runroot = "/run/user/$(id -u)/containers"
graphroot = "$HOME/.local/share/containers/storage"

[storage.options.overlay]
mount_program = "/usr/bin/fuse-overlayfs"
mountopt = "nodev,fsync=0"
EOF
```

Optional: Registries Configuration

```
cat > ~/.config/containers/registries.conf << 'EOF'
unqualified-search-registries = ["registry.access.redhat.com", "registry.redhat.io", "docker.io", "quay.io"]

[[registry]]
location = "docker.io"
insecure = false
EOF
```

---

6. Create Your First Production Quadlet

A) Example: Nginx reverse proxy / web service

```
cat > ~/.config/containers/systemd/nginx.container << 'EOF'
[Unit]
Description=Nginx Web Server (Production)
After=network-online.target
Wants=network-online.target

[Container]
Image=docker.io/library/nginx:alpine
ContainerName=nginx
PublishPort=8080:80
Volume=nginx-data.volume:/usr/share/nginx/html:Z
Volume=nginx-conf.volume:/etc/nginx/conf.d:Z
```

```
AutoUpdate=registry
Environment=TZ=America/New_York
PodmanArgs=--memory=512m --cpus=1.0 --memory-swap=512m

[Service]
Restart=always
TimeoutStartSec=300

[Install]
WantedBy=default.target
EOF
```

A) Create supporting volumes:

```
cat > ~/.config/containers/systemd/nginx-data.volume << 'EOF'
[Volume]
VolumeName=nginx-data
EOF

cat > ~/.config/containers/systemd/nginx-conf.volume << 'EOF'
[Volume]
VolumeName=nginx-conf
EOF
```

B) Another Example

```
cat > ~/.config/containers/systemd/myapp.container << 'EOF'
[Unit]
Description=My Application (Production)
After=network-online.target
Wants=network-online.target

[Container]
Image=your-registry/your-app:latest
ContainerName=myapp
PublishPort=8080:8080
Volume=myapp-data.volume:/data:Z
AutoUpdate=registry
Environment=TZ=America/New_York
PodmanArgs=--memory=1g --cpus=1.5 --memory-swap=1g
```

```
[Service]
Restart=always
TimeoutStartSec=300

[Install]
WantedBy=default.target
EOF
```

B) Create supporting volume:

```
cat > ~/.config/containers/systemd/myapp-data.volume << 'EOF'
[Volume]
VolumeName=myapp-data
EOF
```

---

7. Activate the Service

A) Example for Nginx

```
# Reload systemd user units
systemctl --user daemon-reload

# Start and enable
systemctl --user enable --now nginx.service

# Check status
systemctl --user status nginx.service
podman ps
```

View logs:

```
journalctl --user -u nginx.service -f
```

B) Second Example

```
systemctl --user daemon-reload
systemctl --user enable --now myapp.service

# Verify
systemctl --user status myapp.service
podman ps
```

View logs:

```
journalctl --user -u myapp.service -f
```

---

8. Configure pfSense HAProxy

In pfSense HAProxy:

1. **Backend**
   - Mode: http
   - Server: IP of your RHEL 10 host
   - Port: 8080 (or whatever you published)
   - Health check: recommended
2. **Frontend**
   - Bind to WAN / desired interface
   - TLS offloading (recommended)
   - ACL based on Host header (e.g. hdr(host) -i app.yourdomain.com)
   - Use backend created above
3. **Firewall Rules**
   - Allow traffic from HAProxy (or LAN) to the RHEL host on the published port(s) only.

Firewall example:

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

---

9. Production Hardening Checklist

Item	Command / Action	Status
SELinux	Keep enforcing	☐
Linger enabled	loginctl show-user podman	☐
Resource limits	Set in Quadlet (--memory, --cpus)	☐
Auto-update	AutoUpdate=registry in Quadlet	☐
Firewall	Open only required ports	☐
Log rotation	Configure journald or ship logs	☐
Image scanning	Use podman image scan or external scanner	☐
Backup volumes	Backup ~/.local/share/containers	☐
Non-interactive user	Disable password / use key-based access only	☐
Log management	journalctl --user or forward logs	

Enable auto-update timer (as podman user):

```
systemctl --user enable --now podman-auto-update.timer
```

---

9. Useful Management Commands

A) Example for Nginx

```
# List all user services
systemctl --user list-units --type=service

# Restart a service
systemctl --user restart nginx.service

# Stop a service
systemctl --user stop nginx.service

# View resource usage
podman stats
```

```
# Update all containers with AutoUpdate
podman auto-update
```

B) Another Example

```
# Service management
systemctl --user status myapp.service
systemctl --user restart myapp.service
systemctl --user stop myapp.service

# Container status & resources
podman ps
podman stats

# Logs
journalctl --user -u myapp.service -f

# Manual update
podman auto-update
```

---

10. Optional: Auto-Update Timer

Enable Podman’s auto-update timer (as the podman user):

Bash

```
systemctl --user enable --now podman-auto-update.timer
```

Summary of Key Directories (Rootless)

Purpose	Path
Quadlet files	~/.config/containers/systemd/
Container storage	~/.local/share/containers/storage/
User systemd units	~/.config/systemd/user/

Purpose	Path
Configuration	~/.config/containers/

Summary

- HAProxy on pfSense = edge reverse proxy / TLS / routing
- RHEL 10 + rootless Podman + Quadlet = application runtime
- No Nginx needed unless the application itself requires it

Revision #4
Created 3 August 2026 14:20:22 by Steve Ling
Updated 3 August 2026 16:55:57 by Steve Ling