RHEL STIG RAG - Podman Deployment Guide

Comprehensive guide for deploying the RHEL STIG RAG system using Podman with container orchestration and systemd integration.

Overview

This Podman deployment provides:

- **Rootless Security**: Enhanced security through rootless containers
- **Systemd Integration**: Native systemd service management
- **Pod Orchestration**: Multi-container pod support
- **Auto-recovery**: Automatic restart and health monitoring
- **RHEL-Native**: Optimized for Red Hat environments

Quick Start

Option 1: Automated Script

```bash
# Clone repository
git clone <repository-url>
cd rhel-stig-rag

# Make script executable
chmod +x deploy-podman.sh

# Deploy (rootless mode - default)
./deploy-podman.sh deploy

# Deploy in rootful mode
./deploy-podman.sh --rootful deploy
```

Option 2: Ansible Deployment

```bash
# Deploy with Ansible
cd ansible
ansible-playbook -i inventory/hosts podman-deploy.yml

# Deploy in rootful mode
ansible-playbook -i inventory/hosts podman-deploy.yml -e "stig_rag_podman_mode=rootful"
```

Prerequisites

System Requirements

- **OS**: RHEL 8+, CentOS 8+, Fedora 35+, Ubuntu 20.04+
- **Memory**: 4GB RAM minimum (8GB recommended)
- **Storage**: 5GB free space
- **CPU**: 2 cores minimum

Software Requirements

```bash
# RHEL/CentOS/Fedora
sudo dnf install podman buildah skopeo

# Ubuntu/Debian
sudo apt install podman buildah
```

Deployment Modes

Rootless Mode (Recommended)

Advantages:

- Enhanced security (no root privileges)
- User namespace isolation
- No daemon running as root

Usage:

```bash
# Deploy rootless
./deploy-podman.sh --rootless deploy

# Status check
podman pod ps
systemctl --user status stig-rag-pod
```

Rootful Mode

Advantages:

- Better performance for I/O operations
- Easier networking configuration
- System-wide service management

Usage:

```bash
# Deploy rootful
./deploy-podman.sh --rootful deploy

# Status check
sudo podman pod ps
sudo systemctl status stig-rag-pod
```

Container Architecture

Pod Structure

```
stig-rag-pod
├── stig-rag (main application)
├── stig-rag-metrics (optional monitoring)
└── shared network namespace
```

Volume Mapping

```
Host Directory          → Container Path
~/stig-rag-data/stig_data      → /app/stig_data
~/stig-rag-data/stig_chroma_db → /app/stig_chroma_db
~/stig-rag-logs               → /app/logs
~/stig-rag-config             → /app/config
```

Configuration

Environment Variables

Key configuration options:

```bash
# Application settings
APP_HOST=0.0.0.0
APP_PORT=8000
DEFAULT_RHEL_VERSION=9

# Performance tuning
CHUNK_SIZE=1000
EMBEDDING_MODEL=all-MiniLM-L6-v2
LLM_PROVIDER=huggingface

# Resource limits
MEMORY_LIMIT=4g
CPU_LIMIT=2.0
```

Custom Configuration

Edit the configuration file:

```bash
# Rootless mode
vim ~/stig-rag-config/config.env

# Rootful mode
sudo vim /etc/stig-rag/config.env

# Restart to apply changes
./manage-stig-rag.sh restart
```

Management Commands

Using Management Script

```bash
# Check status
./manage-stig-rag.sh status

# View logs
./manage-stig-rag.sh logs
./manage-stig-rag.sh logs -f  # Follow logs

# Start/stop services
./manage-stig-rag.sh start
./manage-stig-rag.sh stop
./manage-stig-rag.sh restart

# Health check
./manage-stig-rag.sh health

# Update containers
./manage-stig-rag.sh update

# Open shell in container
./manage-stig-rag.sh shell

# Manual backup
./manage-stig-rag.sh backup
```

Using Podman Commands

```bash
# Pod management
podman pod ps
podman pod start stig-rag-pod
podman pod stop stig-rag-pod

# Container management
podman ps
podman logs stig-rag
podman exec -it stig-rag /bin/bash

# Health check
podman healthcheck run stig-rag
```

Using Systemd Commands

Rootless mode:

```bash
systemctl --user status stig-rag-pod
systemctl --user restart stig-rag-pod
systemctl --user enable stig-rag-pod
```

Rootful mode:

```bash
sudo systemctl status stig-rag-pod
sudo systemctl restart stig-rag-pod
sudo systemctl enable stig-rag-pod
```

Advanced Features

Systemd Integration with Quadlet

For systemd 250+ (RHEL 9+), you can use Quadlet for better integration:

```bash
# Enable Quadlet mode
ansible-playbook podman-deploy.yml -e "stig_rag_podman_quadlet_enabled=true"

# Check generated services
systemctl --user list-units | grep stig-rag
```

Auto-Updates

Enable automatic container updates:

```bash
# Enable auto-update
ansible-playbook podman-deploy.yml -e "stig_rag_podman_auto_update=true"

# Check auto-update status
systemctl --user status podman-auto-update.timer
```

Monitoring Integration

Deploy with Prometheus metrics:

```bash
# Deploy with monitoring
ansible-playbook podman-deploy.yml -e "stig_rag_podman_expose_metrics=true"

# Access metrics
curl http://localhost:9090/metrics
```

Backup Management

Automated backups are configured by default:

```bash
# Check backup timer
systemctl --user status stig-rag-backup.timer

# Manual backup
systemctl --user start stig-rag-backup.service

# Restore from backup
tar -xzf ~/stig-rag-backups/stig-rag-backup-*.tar.gz -C ~/stig-rag-data/
```

Networking

Port Configuration

Default ports:

- **8000**: STIG RAG API
- **9090**: Prometheus metrics (if enabled)

Custom Network

The deployment creates a dedicated network:

```bash
# Inspect network
podman network inspect stig-rag-net

# Connect additional containers
podman run --network stig-rag-net <other-container>
```

Security Considerations

SELinux Integration

Podman automatically handles SELinux contexts with the `:Z` flag:

```bash
# Check SELinux contexts
ls -Z ~/stig-rag-data/
```

Security Options

Default security configurations:

- Dropped capabilities: ALL
- Security labels: container_runtime_t
- Read-only root filesystem (optional)
- PID limits

Rootless Benefits

- No root daemon
- User namespace isolation
- Limited filesystem access
- Reduced attack surface

Troubleshooting

Common Issues

1. Permission Denied

Rootless mode issue:

```bash
# Check user namespace
podman unshare cat /proc/self/uid_map

# Fix storage permissions
podman system reset
```

2. Port Already in Use

```bash
# Change port
./deploy-podman.sh --port 8080 deploy

# Or stop conflicting service
sudo ss -tulpn | grep :8000
```

3. Container Won't Start

bash

```
# Check logs
podman logs stig-rag

# Check systemd
systemctl --user status stig-rag-pod -l
journalctl --user -u stig-rag-pod -f
```

4. Health Check Failing

bash

```
# Manual health check
curl -v http://localhost:8000/health

# Check container health
podman inspect stig-rag | jq '.[0].State.Health'
```

Debug Mode

Enable debug logging:

bash

```
# Deploy with debug
ansible-playbook podman-deploy.yml -e "stig_rag_log_level=DEBUG"

# Check debug logs
./manage-stig-rag.sh logs | grep DEBUG
```

Performance Tuning

Resource Limits

Adjust based on your system:

yaml

```
# In vars/podman.yml
stig_rag_podman_memory_limit: "8g"
stig_rag_podman_cpu_limit: "4.0"
stig_rag_podman_pids_limit: 2048
```

Storage Optimization

```bash
# Monitor storage usage
podman system df

# Clean up unused resources
podman system prune -a

# Optimize storage driver
# Edit ~/.config/containers/storage.conf
```

Model Optimization

Use smaller models for better performance:

```yaml
stig_rag_embedding_model: "all-MiniLM-L6-v2"  # Faster, smaller
stig_rag_llm_model: "microsoft/DialoGPT-small"  # Reduced memory
```

Integration Examples

CI/CD Pipeline

```yaml
# .gitlab-ci.yml
deploy:
  script:
    - ansible-playbook ansible/podman-deploy.yml
  only:
    - main
```

Load Balancer

bash

```bash
# Deploy multiple instances
for i in {1..3}; do
  ./deploy-podman.sh --port $((8000 + i)) --pod-name stig-rag-pod-$i deploy
done

# Configure nginx load balancer
# (nginx configuration not included)
```

Development Workflow

bash

```bash
# Development deployment
./deploy-podman.sh --rootless deploy

# Code changes
vim rhel_stig_rag.py

# Rebuild and update
./deploy-podman.sh build
./manage-stig-rag.sh update
```

Migration from Docker

Docker to Podman

bash

```bash
# Convert docker-compose.yml to Podman
podman-compose up -d

# Or use the provided Podman deployment
./deploy-podman.sh deploy
```

Volume Migration

bash

```bash
# Copy data from Docker volumes
docker cp stig-rag-container:/app/stig_data ~/stig-rag-data/
docker cp stig-rag-container:/app/stig_chroma_db ~/stig-rag-data/
```

Maintenance

Regular Updates

```bash
# Update system packages
sudo dnf update podman

# Update container images
./manage-stig-rag.sh update

# Update STIG data
# (automation via ansible playbook)
ansible-playbook update_stig.yml
```

Monitoring Health

```bash
# Check service health
./manage-stig-rag.sh health

# Monitor logs
./manage-stig-rag.sh logs -f

# System resource usage
podman stats stig-rag
```

Backup and Recovery

```bash
# Automated backup (daily)
systemctl --user enable stig-rag-backup.timer

# Manual backup
./manage-stig-rag.sh backup

# Recovery process
./deploy-podman.sh stop
tar -xzf backup.tar.gz -C ~/stig-rag-data/
./deploy-podman.sh start
```

Support and Resources

Useful Commands

```bash
# Complete system info
podman info

# Network information
podman network ls

# Volume information
podman volume ls

# System events
podman events
```

Log Locations

- **Container logs**: `podman logs stig-rag`
- **Systemd logs**: `journalctl --user -u stig-rag-pod`
- **Application logs**: `~/stig-rag-logs/stig_rag.log`

Performance Monitoring

```bash
# Resource usage
podman stats stig-rag

# Container inspection
podman inspect stig-rag

# Health check history
podman healthcheck run stig-rag
```

This Podman deployment provides enterprise-grade container management with enhanced security, better integration with RHEL/systemd, and comprehensive automation through Ansible. The rootless mode is particularly valuable for security-conscious environments while maintaining full functionality.