Local Registry Mirror Setup Guide for RHEL 9

Table of Contents

Prerequisites

Before starting, ensure you have:

- RHEL 9 system with root or sudo access
- Sufficient storage space for container images (recommended: 50GB+)
- Network connectivity to source registries
- Valid SSL certificates (for production use)

Step 1: Install Required Packages

First, install Podman and other necessary tools:

```
# Install podman and related tools
sudo dnf install -y podman httpd-tools skopeo

# Install container registry package
sudo dnf install -y container-tools
```

Step 2: Create Registry Storage Directory

Create a dedicated directory structure for your registry:

```
# Create directories for registry data and certificates
sudo mkdir -p /opt/registry/{data,auth,certs}

# Set appropriate permissions
sudo chown -R $USER:$USER /opt/registry
```

Step 3: Generate SSL Certificates (For Production)

For production environments, generate proper SSL certificates. For testing, you can create self-signed certificates:

```
# Generate self-signed certificate (for testing only)
cd /opt/registry/certs

# Generate private key
openssl genrsa -out registry.key 2048

# Generate certificate signing request
openssl req -new -key registry.key -out registry.csr \
  -subj "/C=US/ST=State/L=City/O=Organization/CN=registry.local"

# Generate self-signed certificate
openssl x509 -req -days 365 -in registry.csr \
  -signkey registry.key -out registry.crt

# For production, use certificates from a trusted CA
```

Step 4: Create Basic Authentication (Optional but Recommended)

Set up basic authentication for your registry:

```
# Create htpasswd file with a user
htpasswd -bBc /opt/registry/auth/htpasswd registryuser registrypassword

# You can add more users with:
# htpasswd -bB /opt/registry/auth/htpasswd anotheruser anotherpassword
```

Step 5: Create Registry Configuration File

Create a configuration file for the registry:

```
cat > /opt/registry/config.yml << 'EOF'
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
auth:
  htpasswd:
    realm: basic-realm
    path: /auth/htpasswd
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
EOF
```

Step 6: Run the Registry Container

Start the registry using Podman:

With Authentication and TLS (Recommended)

```
# Run registry with authentication and TLS
sudo podman run -d \
  --name local-registry \
  --restart=always \
  -p 5000:5000 \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  -v /opt/registry/config.yml:/etc/docker/registry/config.yml:z \
  -e REGISTRY_AUTH=htpasswd \
  -e REGISTRY_AUTH_HTPASSWD_REALM=Registry-Realm \
  -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.crt \
  -e REGISTRY_HTTP_TLS_KEY=/certs/registry.key \
  docker.io/library/registry:2
```

Without TLS (Testing Only)

```
# For testing without TLS (not recommended for production):
sudo podman run -d \
  --name local-registry \
  --restart=always \
  -p 5000:5000 \
  -v /opt/registry/data:/var/lib/registry:z \
  docker.io/library/registry:2
```

Step 7: Configure Firewall

Open the registry port in the firewall:

```
# Add firewall rule
sudo firewall-cmd --permanent --add-port=5000/tcp
sudo firewall-cmd --reload
```

Step 8: Configure Podman to Trust the Registry

For self-signed certificates, configure Podman to trust your registry:

```
# Copy certificate to trusted location
sudo cp /opt/registry/certs/registry.crt /etc/pki/ca-trust/source/anchors/
sudo update-ca-trust

# For insecure registries (testing only), edit registries.conf
sudo vi /etc/containers/registries.conf

# Add your registry to insecure registries section:
# [[registry]]
# location = "localhost:5000"
# insecure = true
```

Step 9: Test Registry Access

Test that you can login to your registry:

```
# Login to registry
podman login localhost:5000 -u registryuser -p registrypassword

# Test pushing an image
podman pull docker.io/library/alpine:latest
podman tag alpine:latest localhost:5000/alpine:latest
podman push localhost:5000/alpine:latest
```

Copying Images from External Registries

Method 1: Using Skopeo (Recommended)

Skopeo is the most efficient tool for copying images between registries:

```
# Copy single image from Red Hat Quay to local registry
skopeo copy \
  --src-creds=source_username:source_password \
  --dest-creds=registryuser:registrypassword \
  docker://quay.io/organization/image:tag \
  docker://localhost:5000/organization/image:tag

# Copy with all tags (requires skopeo >= 1.0)
skopeo copy \
  --all \
  --src-creds=source_username:source_password \
  --dest-creds=registryuser:registrypassword \
  docker://quay.io/organization/image \
  docker://localhost:5000/organization/image
```

Method 2: Using Podman Pull and Push

For individual images using Podman:

```
# Login to source registry
podman login quay.io

# Pull image
podman pull quay.io/organization/image:tag

# Tag for local registry
podman tag quay.io/organization/image:tag localhost:5000/organization/image:tag

# Push to local registry
podman push localhost:5000/organization/image:tag
```

Method 3: Bulk Mirror Script

Create a script for bulk mirroring:

```
cat > /opt/registry/mirror-images.sh << 'EOF'
#!/bin/bash

# Configuration
SOURCE_REGISTRY="quay.io"
DEST_REGISTRY="localhost:5000"
SOURCE_CREDS="source_user:source_pass"
DEST_CREDS="registryuser:registrypassword"

# List of images to mirror
IMAGES=(
  "organization/image1:v1.0"
  "organization/image2:latest"
  "organization/image3:v2.1"
)

# Mirror each image
for IMAGE in "${IMAGES[@]}"; do
  echo "Mirroring $IMAGE..."
  skopeo copy \
    --src-creds="$SOURCE_CREDS" \
    --dest-creds="$DEST_CREDS" \
    "docker://${SOURCE_REGISTRY}/${IMAGE}" \
    "docker://${DEST_REGISTRY}/${IMAGE}"

  if [ $? -eq 0 ]; then
    echo "Successfully mirrored $IMAGE"
  else
    echo "Failed to mirror $IMAGE"
  fi
done
EOF

chmod +x /opt/registry/mirror-images.sh
```

Step 10: Set Up Automatic Mirroring (Optional)

Create a systemd service for periodic mirroring:

Create Systemd Service

```
# Create systemd service
cat > /etc/systemd/system/registry-mirror.service << 'EOF'
[Unit]
Description=Registry Mirror Service
After=network.target

[Service]
Type=oneshot
ExecStart=/opt/registry/mirror-images.sh
User=root

[Install]
WantedBy=multi-user.target
EOF
```

Create Timer for Periodic Execution

```
# Create timer for periodic execution
cat > /etc/systemd/system/registry-mirror.timer << 'EOF'
[Unit]
Description=Run Registry Mirror Service every 6 hours
Requires=registry-mirror.service

[Timer]
OnCalendar=*-*-* 00,06,12,18:00:00
Persistent=true

[Install]
WantedBy=timers.target
EOF

# Enable and start timer
sudo systemctl daemon-reload
sudo systemctl enable --now registry-mirror.timer
```

Registry Management and Maintenance

Viewing Registry Contents

List images in your local registry:

```
# Using curl (if registry API is accessible)
curl -u registryuser:registrypassword https://localhost:5000/v2/_catalog

# List tags for specific image
curl -u registryuser:registrypassword https://localhost:5000/v2/organization/image/tags/list
```

Registry Garbage Collection

Clean up unused layers:

```
# Stop registry
sudo podman stop local-registry

# Run garbage collection
sudo podman run --rm \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/config.yml:/etc/docker/registry/config.yml:z \
  docker.io/library/registry:2 \
  registry garbage-collect /etc/docker/registry/config.yml

# Restart registry
sudo podman start local-registry
```

Monitoring Registry

Check registry logs and status:

```
# View logs
sudo podman logs local-registry

# Check container status
sudo podman ps -a | grep local-registry

# Check disk usage
du -sh /opt/registry/data
```

Troubleshooting Tips

Common Issues and Solutions

1. **Certificate Problems**
   - Ensure certificates are properly configured and trusted by the system
   - Verify certificate paths in the podman run command
   - Check certificate validity: `openssl x509 -in /opt/registry/certs/registry.crt -text -noout`
2. **Authentication Failures**
   - Verify htpasswd file permissions: `ls -l /opt/registry/auth/htpasswd`
   - Test credentials manually: `htpasswd -v /opt/registry/auth/htpasswd registryuser`
   - Check registry logs for authentication errors
3. **Push/Pull Errors**
   - Check firewall rules: `sudo firewall-cmd --list-all`
   - Verify registry is running: `sudo podman ps | grep local-registry`
   - Test connectivity: `curl -k https://localhost:5000/v2/`
4. **Storage Issues**
   - Monitor disk space: `df -h /opt/registry/data`
   - Check for permission issues: `ls -la /opt/registry/`
   - Review registry logs for storage errors
5. **Network Issues**
   - Verify connectivity to source registries: `curl -I https://quay.io`
   - Check DNS resolution: `nslookup quay.io`
   - Test with wget or curl before using skopeo

Red Hat Specific Considerations

For Red Hat-specific registries, ensure you have:

- Valid Red Hat subscription
- Proper authentication tokens or service accounts
- Pull secrets for accessing registry.redhat.io

```
# Login to Red Hat registry
podman login registry.redhat.io
```

Security Best Practices

1. **Always use TLS in production**
   - Use certificates from a trusted CA
   - Implement certificate rotation policies
2. **Implement strong authentication**
   - Use complex passwords
   - Consider implementing token-based authentication
   - Regularly rotate credentials
3. **Network Security**
   - Restrict registry access to specific networks
   - Use firewall rules to limit access
   - Consider implementing a reverse proxy
4. **Regular Updates**
   - Keep registry image updated
   - Update RHEL 9 system regularly
   - Monitor security advisories
5. **Backup Strategy**
   - Regular backup of `/opt/registry/data`
   - Document registry configuration
   - Test restore procedures

Conclusion

This setup provides you with a fully functional local registry mirror that can cache and serve container images locally, reducing bandwidth usage and improving deployment speeds in your RHEL 9 environment. The registry is configured with security best practices and can be easily integrated into your container workflow.