Running PostgreSQL with Podman, Quadlets & Systemd

Dirk Aumueller
d.aumueller@proventa.de
Proventa AG

2025-12-15

PROVENTA

Agenda

- Let’s Face the Reality in Germany
- sytemd
- Podman
- Quadlets
- Lingering
- Simple Example
- Future Outlook

Introduction

- Dirk Aumueller
- Associate Partner @ Proventa AG
- Consulting & Support
- Wide customer spectrum
  - SMEs
  - Enterprises
  - Government Agencies

Let's Face the Reality in Germany

- Containers are a commodity
- Kubernetes is complex
  - Companies still lack trained employees
- Virtual machines are still the main deployment target
- Security creates difficult environments for deployments

How can we team up containers and Linux services in a VM to run PostgreSQL?

Requirements

- Strict separation of OS and application (= DB)
- No root privileges
- Standardized deployments
- High update / upgrade velocity
- Automation friendly

systemd

- systemd is for unified system & service management on Linux
- Nearly all Linux distributions ship systemd as default
- Main component is an init system
  - Bootstraps user space
  - Manages user processes
- Integrates additional functionalities
  - Device management
  - Login management
  - Network connection management
  - Event logging

```
dca@vbox:~$ systemctl [start|stop|reload|restart|status] postgresql.service
```

systemd Unitfile

- Unitfile is a plain text ini-style file
- Encodes information about objects controlled & supervised by systemd
  - Service
  - Socket
  - Device
  - (Auto)mount point
  - Swap file / partition
  - Start-up target
  - Timer
  - Resource management slice
  - Group of externally created processes

systemd Unitfile Example

```
[Unit]
Description=Postgres Server
Requires=docker.service
After=docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker run \
            --rm --name postgres --net=web \
            -e POSTGRES_PASSWORD=$POSTGRES_PASSWORD \
            -v /opt/postgres:/var/lib/postgresql/data \
            -m 768M \
            -p 5432:5432 \
            postgres:18.1
ExecStop=/usr/bin/docker stop postgres

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

Podman

- Open source tool to manage, develop and execute containers
- Daemonless
- Rootless containers allow you to contain privileges
- Podman Desktop available to manage all your containers
  - Regardless of container engine
- Compatible with other OCI compliant container formats including Docker
- Requires Cgroups v2

```
1  dca@vbox:~$ podman info --format {{.Host.CgroupsVersion}}
2  v2
```

Podman Command Examples

```
# Pull the Postgres container image
podman pull docker.io/postgres:18.1

# List local images
podman images

# Run a Postgres container
podman run --name pg18 -e POSTGRES_PASSWORD=mysecret -d postgres:18.1

# List running containers
podman ps

# Stop the Postgres container
podman stop pg18

# Remove the container
podman rm pg18
```

Quadlets

- Podman always had impressive integration with systemd
  - `podman generate systemd`
- Managing dependencies, networks & volumes was somewhat problematic
  - Required manual work
- Declarative approach for container setups, dependencies, networks & volumes with files
- Podman & systemd take care of service file creation
- Quadlets make management of containers easy & more maintainable
  - Comparable to Docker Compose

> Quadlets are available since Podman v4.4, but I recommend at least v5.

Quadlet Filetypes

Filetype	Description
`.container`	Container configuration
`.volume`	Volume definition, e.g. local or S3
`.network`	Network definition for containers or pods
`.pod`	Pod definition
`.build`	Build instructions for a container image
`.kube`	Executes k8s manifests as a service
`.image`	Describes the image pull command

Lingering

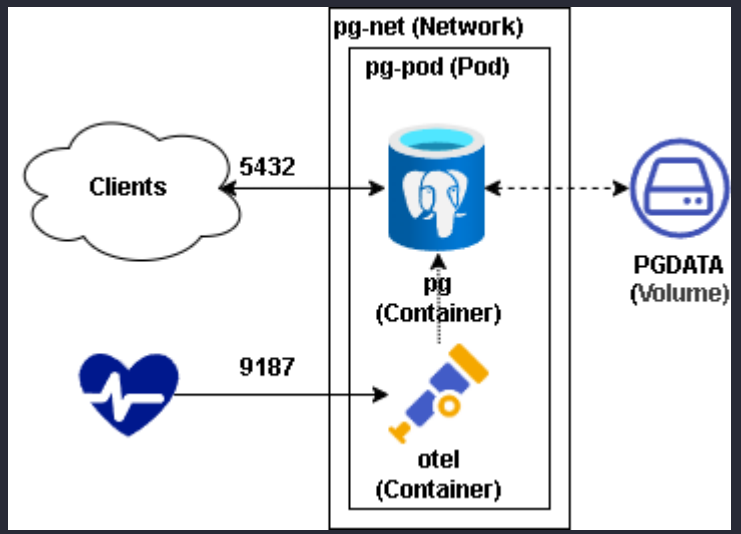
- Run containers as unprivileged user without login session
- The systemd login manager will terminate any user services when their login session ends
- Use `loginctl` with `enable-linger` option to configure a user to start during the boot process

Activate Lingering

```
 1  dca@vbox:~$ loginctl list-users
 2   UID USER LINGER STATE
 3  1000 dca  no     active
 4
 5  1 users listed.
 6
 7  dca@vbox:~$ sudo loginctl enable-linger dca
 8
 9  dca@vbox:~$ loginctl list-users
10   UID USER LINGER STATE
11  1000 dca  yes    active
12
13  1 users listed.
```

Simple Example

- Setup a PostgreSQL v18 container
  - Persist data directory on a volume
- Add an opentelemetry container to publish Postgres metrics
- Run both containers in a pod
- Use a dedicated network



Volume- & Network-Quadlet

~/.config/containers/systemd/pg-data.volume

```
1 [Unit]
2 Description=Volume for PostgreSQL data directory
3
4 [Volume]
5 Label=app=postgres
6 VolumeName=pg-data
```

~/.config/containers/systemd/pg.network

```
1 [Unit]
2 Description=Dedicated network for all containers related to PostgreSQL
3
4 [Network]
5 Label=app=postgres
6 NetworkName=pg
```

Pod Configuration

~/.config/containers/systemd/pg.pod

```
1 [Unit]
2 Description=Pod for PostgreSQL & opentelemetry
3
4 [Pod]
5 PodName=pg
6 Network=pg.network
7 PublishPort=5432:5432
8 PublishPort=9187:9187
```

Podman Secret for the Superuser Password

```
1  dca@vbox:~$ printf 'postgrespw' | podman secret create pg_postgres_password -
2  71903c25e80e0accb9ca8bf88
3
4  dca@vbox:~$ podman secret ls
5  ID                         NAME                  DRIVER      CREATED         UPDATE
6  71903c25e80e0accb9ca8bf88  pg_postgres_password  file        22 seconds ago  22 sec
```

Container-Quadlet for PostgreSQL

~/.config/containers/systemd/pg-db.container

```
[Unit]
Description=PostgreSQL Container
After=network-online.target

[Container]
Label=app=postgres
ContainerName=pg-db
Image=docker.io/postgres:18.1
Pod=pg.pod
Volume=pg-data.volume:/var/lib/postgresql
AutoUpdate=registry
Environment=TZ=Europe/Berlin
Secret=pg_postgres_password,type=env,target=POSTGRES_PASSWORD

[Service]
Restart=always

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

opentelemetry Configuration

~/otel_config.yaml

```yaml
receivers:
  postgresql:
    endpoint: localhost:5432
    transport: tcp
    username: otel
    password: otelpw
    databases:
      - postgres
    collection_interval: 10s
    tls:
      insecure: true
      insecure_skip_verify: true

processors:
  batch:
    timeout: 10s

exporters:
  prometheus:
    endpoint: 0.0.0.0:9187

service:
  pipelines:
    metrics:
      receivers: [postgresql]
```

Container-Quadlet for opentelemetry

~/.config/containers/systemd/pg-otel.container

```
1  [Unit]
2  Description=opentelemetry Container
3  After=network-online.target
4
5  [Container]
6  Label=app=postgres
7  ContainerName=pg-otel
8  Image=docker.io/otel/opentelemetry-collector-contrib:0.141.0
9  Pod=pg.pod
10 AutoUpdate=registry
11 Environment=TZ=Europe/Berlin
12 Volume=%h/otel_config.yaml:/etc/otelcol-contrib/config.yaml:Z
13
14 [Service]
15 Restart=always
16
17 [Install]
18 WantedBy=multi-user.target default.target
```

Postgres User for opentelemetry

~/pg_add_otel_user.sql

```
1  CREATE EXTENSION IF NOT EXISTS pg_stat_statements;
2  CREATE USER otel WITH PASSWORD 'otelpw';
3  GRANT pg_monitor TO otel;
```

- Mount script into Postgres container by modifying `postgres.container` file

```
1  Volume=%h/pg_add_otel_user.sql:/docker-entrypoint-initdb.d/pg_add_otel_user.sql
```

Automatic Container Updates

- Podman ships with a systemd timer that checks for container updates
  - Runs daily by default, if enabled
- Checks all containers with `AutoUpdate=registry`
- Pulls newer image, if available
- Restarts containers with new image
- Keeps old image in case you need to roll back

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

# Check for next update run
systemctl --user list-timers podman-auto-update.timer
```

Pod Deployment

- Use the Quadlet generator to create all services

```
1  systemctl --user daemon-reload
```

- Pod creation
- Container creation
- Link containers to pod
- Set up dependencies
- Enable all services to start at boot

Check Podman Status

```
dca@vbox:~$ podman volume ls
DRIVER      VOLUME NAME
local       pg-data

dca@vbox:~$ podman network ls
NETWORK ID    NAME        DRIVER
c936e3b569d9  pg          bridge
2f259bab93aa  podman      bridge

dca@vbox:~$ podman ps
CONTAINER ID  IMAGE                                          COMMAND
f8edd42fe14c
d4d48515dbde  docker.io/library/postgres:18.1                postgres
a0c70260118d  docker.io/otel/opentelemetry-collector-contrib:0.141.0  --config /etc
```

Check Postgres & opentelemetry

```
dca@vbox:~$ podman exec -it pg-db psql -U postgres -c "SELECT version();"
                                                 version
-------------------------------------------------------------------------------
 PostgreSQL 18.1 (Debian 18.1-1.pgdg13+2) on x86_64-pc-linux-gnu, compiled by gcc (
(1 row)

dca@vbox:~$ curl "http://localhost:9187/metrics"
# HELP postgresql_backends The number of backends.
# TYPE postgresql_backends gauge
postgresql_backends{otel_scope_name="github.com/open-telemetry/opentelemetry-collec
# HELP postgresql_bgwriter_buffers_allocated_total Number of buffers allocated.
# TYPE postgresql_bgwriter_buffers_allocated_total counter
postgresql_bgwriter_buffers_allocated_total{otel_scope_name="github.com/open-teleme
...
```

Future Outlook

- Automation (with Ansible?)
- Backup & recovery
- High-availability
- Security & Hardening
- Extensions

Thank you! - Questions?