Notes & Labs for EX188 Prep

Exam Objectives:

Implement images using Podman

- Understand and use FROM (the concept of a base image) instruction.
- Understand and use RUN instruction.
- Understand and use ADD instruction.
- Understand and use COPY instruction.
- Understand the difference between ADD and COPY instructions.
- Understand and use WORKDIR and USER instructions.
- Understand security-related topics.
- Understand the differences and applicability of CMD vs. ENTRYPOINT instructions.
- Understand ENTRYPOINT instruction with param.
- Understand when and how to expose ports from a Containerfile.
- Understand and use environment variables inside images.
- Understand ENV instruction.
- Understand container volume.
- Mount a host directory as a data volume.
- Understand security and permissions requirements related to this approach.
- Understand the lifecycle and cleanup requirements of this approach.

Manage images

- Understand private registry security.
- Interact with many different registries.
- Understand and use image tags
- Push and pull images from and to registries.
- Back up an image with its layers and meta data vs. backup a container state.

Run containers locally using Podman

- Run containers locally using Podman
- Get container logs.
- Listen to container events on the container host.
- Use Podman inspect.
- Specifying environment parameters.
- Expose public applications.
- Get application logs.
- Inspect running applications.

Run multi-container applications with Podman

- Create application stacks
- Understand container dependencies
- Working with environment variables
- Working with secrets
- Working with volumes
- Working with configuration

Troubleshoot containerized applications

- Understand the description of application resources
- Get application logs
- Inspect running applications
- Connecting to running containers

PODMAN

Usage

Basic Syntax:

```
podman [OPTIONS]... [COMMAND]
```

Image & Registry Operations:

```
podman login [-u USER] [-p PASS] REGISTRY
podman logout REGISTRY|--all
podman images [-q]
podman rmi [-f] IMAGE...
podman search [--limit LIMIT] [--list-tags] KEYWORD
podman pull [--all-tags] SOURCE
podman tag IMAGE:TAG TARGET_IMAGE:TAG
podman push IMAGE [DESTINATION]
podman build [--squash|squash-all] [--tag IMAGE] PATH
```

Note:

When a user logs in to a registry, an authentication file will be created in **$XDG_RUNTIME_DIR/containers/auth.json**. The user and password is encoded in Base64.

Container Operations:

```
podman run [-d] [--rm] [-it]
      [--name NAME]
      [-p PORT_INFO]
      [-v VOL_INFO]...
      [-e ENV_INFO]...
      IMAGE [CMD]
podman exec [-it] CONTAINER CMD
podman ps [-aq]
podman rm CONTAINER...
podman kill [-s SIGNAL] CONTAINER...
podman start|stop|restart CONTAINER...
podman logs CONTAINER
podman cp SRC DEST
```

Where,

```
SRC = DEST = [CONTAINER:]PATH
```

IMAGE

Building

There are 4 RH images that we can use as a base to build our containers:

ubi	Standard	- dnf, systemd, and utils such as gzip & tar.
ubi-init	Init	- multi-application and systemd.
ubi-minimal	Minimal	- smaller than init but only with microdnf.
ubi-micro	Micro	- smallest image with bare minimum. No pkg mgmt.

Containerfile

```
FROM registry.access.redhat.com/ubi8/ubi:8.7

ARG CUSTOM_BUILD_VAR

ENV DOCUMENT_ROOT=${CUSTOM_BUILD_VAR:-/var/www/html} \
    HTTPD_PORT=8080

LABEL description="ubi with network tools"

EXPOSE $HTTPD_PORT

RUN yum -y install httpd && \
    yum -y clean all && \
    sed -i 's/^Listen 80 *$/Listen '$HTTPD_PORT'/' /etc/httpd/conf/httpd.conf

COPY ./src/ $DOCUMENT_ROOT

WORKDIR $DOCUMENT_ROOT

#USER apache

CMD httpd -DFOREGROUND
```

COPY - copies local files to image
ADD - copies local/remote(URL) file to image
ENTRYPOINT - cannot be overridden
CMD - can be overridden
ARG - build argument used in
`podman build --build-arg CUSTOM_BUILD_VAR=/srv …`

ENTRYPOINT and CMD accept text array format ["CMD", "param1", "param2"] or string format as in the example above.

Multi-stage build for security

```
# syntax=docker/dockerfile:1

FROM ubi AS base
WORKDIR /opt/somedir
RUN compile-somedir

FROM ubi-minimal AS stage1
WORKDIR /opt/appl
COPY --from=builder /opt/somedir/compiled-data ./
CMD ["./app"]
```

podman build --target stage1 -t abc/href-counter:latest .

Management

skopeo inspect|delete|list-tags IMAGE

skopeo copy [OPTIONS]... SRC_IMAGE DEST_IMAGE

Where,

```
IMAGE = SRC_IMAGE
      = DEST_IMAGE
      = containers-storage|dir|docker|oci|tarball:PATH
OPTIONS,
      --src-cred USERNAME[:PASSWORD]
      --dest-cred USERNAME[:PASSWORD]
      --src-username STRING
      --src-password  STRING
      --dest-username STRING
      --dest-password STRING
```

STORAGE

Local Directory

To mount from local directory use -v option for the run command

podman run -v *LOCAL_DIR*:*CONT_DIR*:z|Z *IMAGE*

Where,

z: multiple containers share access to bind mount

Z: container has exclusive access to bind mount

One potential problem with using local directory binding is, the process in the container may need to run as a specific UID. The following method can be used to check what UID and GID the process inside the container will run as:

```
podman run --rm IMAGE id
```

Next, we check and set our local directory permission, to match the containers process. You might need to use *chown*, *chgrp* and *chmod* to perform that operation.

```
podman unshare ls -l /srv/website
podman unshare chgrp -R 123 /srv/website
```

You might need to use chown, chgrp and chmod command to perform the operation.

```
chown [-R] USER[.GROUP] PATH
chgrp [-R] GROUP PATH
chmod PERM PATH
```

For more information, refer to man page **chown(1)**, **chgrp(1)** and **chmod(1)**.

Volume

Volumes let Podman manage the data mounts. Podman will help us configure the SELinux permissions too.

To create a volume myvol, inspect it and make use of the volume in a container:

```
podman volume create myvol
podman volume inspect myvol
podman run --rm -it --volume myvol:/somedir ubi /bin/bash
```

To import from and export to an archived file:

```
podman volume import your_vol data.tar.gz
podman volume export myvol --output backup.tar.gz
podman run --rm --mount
    'type=volume,source=myvol,destination=/somedir,ro'
    ubi /bin/bash
```

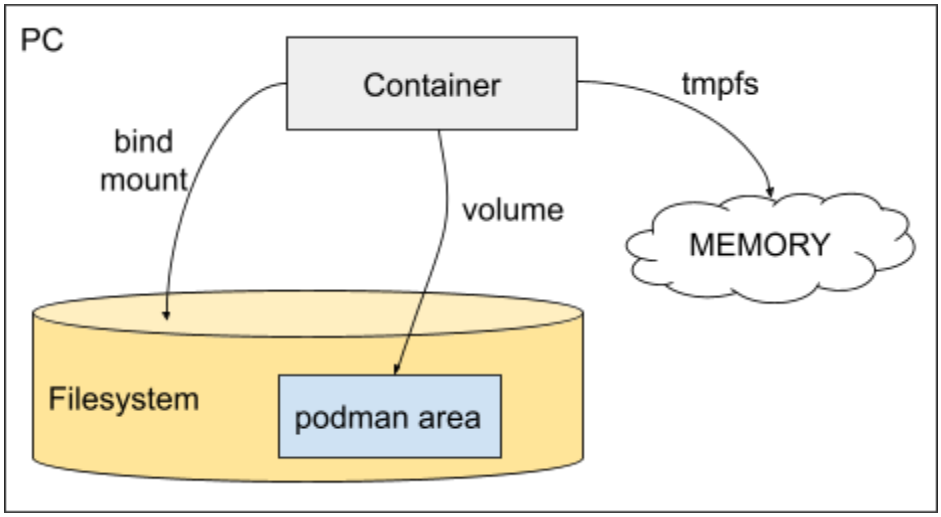

Diagram 1: bind mount vs volume vs tmpfs mount

Using *tmpfs* allows faster access to a non-persistent directory without COW.

```
podman run --mount
    type=tmpfs,tmpfs-size=100M,destination=/var/lib/pgsql/data
    -e  POSTGRESQL_ADMIN_PASSWORD=abc postgresql-13
```

Network

Podman will use the default network named podman for root user. For normal users, podman creates an isolated user network namespace using slirp4netns.

Create network:

```
podman network create NETWORK
```

List all network:

```
podman network ls
```

Inspect network:

```
podman network inspect NETWORK
```

Delete network:

```
podman network rm NETWORK
```

Remove unused network (except default network, podman):

```
podman network prune
```

Add/remove container to a network:

```
podman network [dis]connect NETWORK CONTAINER
```

DNS is disabled in the default “podman” network. That is why we are not able to use dns names when a container is created by the root user. Use *`podman network inspect`* to check if dns_enabled is set to true.

Lab Network:

```
[user@ora ~]$ podman network ls
NETWORK ID    NAME        DRIVER
2f259bab93aa  podman      bridge
[user@ora ~]$ podman network create my-net
my-net
[user@ora ~]$ podman network create your-net
your-net
[user@ora ~]$ podman network inspect podman | grep dns
        "dns_enabled": false,
[user@ora ~]$ podman network inspect my-net
[
        {
        "name": "my-net",
        "id": "abee63a897af09e986af1b00256a490e726cfc637517e01ac1cc5ce95b86f128",
        "driver": "bridge",
        "network_interface": "podman1",
        "created": "2023-07-14T14:15:21.38856544-07:00",
        "subnets": [
                {
                        "subnet": "10.89.0.0/24",
                        "gateway": "10.89.0.1"
                }
        ],
        "ipv6_enabled": false,
        "internal": false,
        "dns_enabled": true,
        "ipam_options": {
                "driver": "host-local"
        }
        }
]
[user@ora ~]$ podman network inspect your-net
[
        {
        "name": "your-net",
        "id": "a52a2609d684a1140576ffd0e06eb4bf56eca5c8fc669dcbbb1b001754448c7a",
        "driver": "bridge",
        "network_interface": "podman2",
        "created": "2023-07-14T14:15:26.594459898-07:00",
        "subnets": [
                {
                        "subnet": "10.89.1.0/24",
                        "gateway": "10.89.1.1"
                }
        ],
        "ipv6_enabled": false,
        "internal": false,
        "dns_enabled": true,
        "ipam_options": {
                "driver": "host-local"
        }
        }
]
```

```
[user@ora ~]$ podman pull registry.access.redhat.com/ubi8/httpd-24
[user@ora ~]$ podman run -d --rm --name my-web --net my-net httpd-24
6dad3dfe8d73a5378aafc8c57b08d5c23b1f73961e0bfe81f1d54a20c732c5be
```

Create some content for the webserver to display:

```
[user@ora ~]$ podman exec -it my-web vi /var/www/html/index.html
[user@ora ~]$ podman exec -it my-web cat /var/www/html/index.html
hello world
```

```
[user@ora ~]$ podman inspect my-web | grep -i expose
                      "io.openshift.expose-services": "8080:http,8443:https",
[user@ora ~]$ podman inspect my-web | grep -i ipaddress
               "IPAddress": "",
                      "IPAddress": "10.89.0.5",
[user@ora ~]$ podman exec -t my-web curl localhost:8080
hello world

[user@ora ~]$ podman run -d --rm --name your-app --net your-net
                 registry.access.redhat.com/ubi8/ubi sleep infinity
1c8e26e40c177d981f9e44898da0bafd8ca69d808e34ef8df895990b4900b576
[user@ora ~]$ podman exec -t your-app curl my-web:8080
curl: (6) Could not resolve host: my-web
[user@ora ~]$ podman network connect my-net your-app
[user@ora ~]$ podman exec -t your-app curl my-web:8080
hello world
[user@ora ~]$ podman inspect your-app | grep -i ipaddress
               "IPAddress": "",
                      "IPAddress": "10.89.0.6",
                      "IPAddress": "10.89.1.4",
[user@ora ~]$ podman network disconnect my-net your-app
[user@ora ~]$ podman exec -t your-app curl my-net:8080
curl: (6) Could not resolve host: my-net
[user@ora ~]$ podman inspect your-app | grep -i ipaddress
               "IPAddress": "",
                      "IPAddress": "10.89.1.4",
[user@ora ~]$ podman stop my-web
my-web
[user@ora ~]$ podman stop your-app
WARN[0010] StopSignal SIGTERM failed to stop container your-app in 10 seconds, resorting to
SIGKILL
Your-app
```

podman-compose

```
[user@WinDev2306Eval ~]$ mkdir stack
[user@WinDev2306Eval ~]$ cd stack
[user@WinDev2306Eval stack]$ vi compose.yml
[user@WinDev2306Eval stack]$ cat compose.yml
services:
  db:
    image: docker.io/library/postgres
    container_name: local_db
    ports:
      - "54320:5432"
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: admin
    volumes:
      - local_db_vol:/var/lib/postgresql/data
    networks:
      - app-net
  pgadmin:
    image: dpage/pgadmin4
    container_name: local_pgadmin
    ports:
      - "5050:80"
    environment:
      PGADMIN_DEFAULT_EMAIL: yourname@email.com
      PGADMIN_DEFAULT_PASSWORD: admin
    volumes:
      - local_pgadmin_vol:/var/lib/pgadmin
    networks:
      - app-net
volumes:
  Local_db_vol:
  local_pgadmin_vol:
networks:
  app-net:
    name: app-net
[user@WinDev2306Eval stack]$ podman-compose up -d
podman-compose version: 1.0.6
['podman', '--version', '']
using podman version: 4.4.1
** excluding:  set()
['podman', 'ps', '--filter', 'label=io.podman.compose.project=stack', '-a',
'--format', '{{ index .Labels "io.podman.compose.config-hash"}}']
…
```

Visit localhost:5050. Try to add the database that was created (local_db), into the web console to see if pgadmin4 can detect your database.

When you are done, try to destroy and check the result. (container, volume, network)

```
[user@WinDev2306Eval stack]$ podman-compose down
podman-compose version: 1.0.6
['podman', '--version', '']
using podman version: 4.4.1
** excluding:  set()
podman stop -t 10 local_pgadmin
local_pgadmin
exit code: 0
podman stop -t 10 local_db
local_db
exit code: 0
podman rm local_pgadmin
local_pgadmin
exit code: 0
podman rm local_db
local_db
exit code: 0

[user@WinDev2306Eval stack]$
```

Troubleshooting Challenge

Create a stack with the following conditions:

1. A backend server “backend” using quay.io/kelvinlai/myserver, connected to network “tmp-net”
2. A frontend server “frontend” using quay.io/kelvinlai/appserver, connected to network “tmp-net”
3. Troubleshoot the containers, so that when you set it up correctly, “podman logs backend” will display the message “Congratulations you managed to solve the problem”

```
[user@WinDev2306Eval stack]$ cat solution.yml
services:
  backend:
    image: quay.io/kelvinlai/myserver
    container_name: backend
    networks:
      - tmp-net
  frontend:
    image: quay.io/kelvinlai/appserver
    container_name: frontend
    environment:
      SERVER_PASS: ABC
      SERVER_PORT: 1234
      SERVER_NAME: backend
    networks:
      - tmp-net
networks:
  tmp-net:
    name: tmp-net

$ podman-compose -f solution.yml up -d
```

LAB 1

1. Installing Podman (Optional)

```
sudo yum install -y podman
```

2. Make sure registry.access.redhat.com is searched first before docker.io. Configure docker.io as an insecure registry:

```
sudo vi /etc/containers/registries.conf
[registries.search]
registries = [‘registry.access.redhat.com’, ‘docker.io’]
[registries.insecure]
registries = [‘docker.io’]
```

   Note:
   - some registries requires you to specify port number to eg: reg.example.com:5000
   - The above is V1 file format. Refer to **container-registries.conf(5)** for V2 format.

3. Create a web server container systemd service named super-web with the following conditions:
   a. Bind the local port 12345 to container port 8080.
   b. Bind the /srv/website local directory to /var/www/html directory inside the container.
   c. Create an index.html in /srv/website with the following content:
      Hello World!
   d. Name the container super-web
   e. The container must be run detached

   Solution:

```
[user@ora User]$ cd
[user@ora ~]$ podman search registry.access.redhat.com/httpd
NAME
DESCRIPTION
registry.access.redhat.com/rhscl/httpd-24-rhel7                                Apache
HTTP 2.4 Server
registry.access.redhat.com/ubi8/httpd-24
Platform for running Apache httpd 2.4 or bui...

[user@ora ~]$ podman pull registry.access.redhat.com/ubi8/httpd-24
Trying to pull registry.access.redhat.com/ubi8/httpd-24:latest...
Getting image source signatures
Copying blob 6c53be4efe39 done
…
Writing manifest to image destination
Storing signatures
1398b5b376eab2e0c4a287fc370f08822c59f62a3b21fc11d9fdf77a2965bb8f
[user@ora ~]$ podman images
REPOSITORY                              TAG          IMAGE ID     CREATED      SIZE
registry.access.redhat.com/ubi8/httpd-24  latest     1398b5b376ea  3 weeks ago  454
MB
```

Extra: if you already know the fully qualified name of the image you could just skip the above step

```
[user@ora ~]$ echo 'Hello World!' | sudo tee /srv/website/index.html
Hello World!
[user@ora ~]$ sudo semanage fcontext -a -t container_file_t "/srv/website(/.*)?"
[user@ora ~]$ restorecon -R -v /srv/website

[user@ora ~]$ podman run -d --name super-web -p 12345:8080 -v
/srv/website:/var/www/html:Z httpd-24
933d6009ca7a4364539acbecf3a4381bf3856678c099b13cbe70966ec2c2b9e2
```

Note: **Use fully qualified image name if it wasn't downloaded previously**

```
[user@ora ~]$ podman logs super-web
=> sourcing 10-set-mpm.sh ...
=> sourcing 20-copy-config.sh ...
=> sourcing 40-ssl-certs.sh ...
---> Generating SSL key pair for httpd...
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.0.2.100. Set the 'ServerName' directive globally to suppress this
message
[Fri Jul 14 15:02:43.589925 2023] [ssl:warn] [pid 1:tid 140538640952768] AH01909:
10.0.2.100:8443:0 server certificate does NOT include an ID which matches the server
name
…
[Fri Jul 14 15:02:43.669917 2023] [core:notice] [pid 1:tid 140538640952768] AH00094:
Command line: 'httpd -D FOREGROUND'
[user@ora ~]$ curl localhost:12345
Hello World!
[user@ora ~]$

[user@ora ~]$ podman generate systemd -n -f super-web
/home/user/container-super-web.service
[user@ora ~]$ mkdir -p ~/.config/systemd/user
[user@ora ~]$ mv container-super-web.service .config/systemd/user/super-web.service
```

```
[user@ora ~]$ systemctl --user daemon-reload
[user@ora ~]$ systemctl --user status super-web.service
○ super-web.service - Podman container-super-web.service
      Loaded: loaded (/home/user/.config/systemd/user/super-web.service; disabled;
vendor preset: disabled)
      Active: inactive (dead)
      Docs: man:podman-generate-systemd(1)
[user@ora ~]$ systemctl --user start super-web.service
Job for super-web.service failed because the service did not take the steps required
by its unit configuration.
See "systemctl --user status super-web.service" and "journalctl --user -xeu
super-web.service" for details.
[user@ora ~]$ systemctl --user status super-web.service
● super-web.service - Podman container-super-web.service
      Loaded: loaded (/home/user/.config/systemd/user/super-web.service; disabled;
vendor preset: disabled)
      Active: active (running) since Fri 2023-07-14 08:58:28 PDT; 7s ago
      Docs: man:podman-generate-systemd(1)
      Process: 21513 ExecStart=/usr/bin/podman start super-web (code=exited,
status=0/SUCCESS)
   Main PID: 21538 (conmon)
      CGroup:
/user.slice/user-1000.slice/user@1000.service/app.slice/super-web.service
             ├─21522 /usr/bin/slirp4netns --disable-host-loopback --mtu=65520
--enable-sandbox --enable-seccomp --enabl>
             ├─21524 rootlessport
             ├─21530 rootlessport-child
             ├─21538 /usr/bin/conmon --api-version 1 -c
933d6009ca7a4364539acbecf3a4381bf3856678c099b13cbe70966ec2c2b9e>
             ├─21541 httpd -D FOREGROUND
             ├─21569 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21570 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21571 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21572 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21573 httpd -D FOREGROUND
             ├─21574 httpd -D FOREGROUND
             ├─21575 httpd -D FOREGROUND
             └─21576 httpd -D FOREGROUND

Jul 14 08:58:28 ora super-web[21538]: [Fri Jul 14 15:58:28.915036 2023] [:notice]
[pid 1:tid 140035002568128] ModSecuri>
Jul 14 08:58:28 ora super-web[21538]: [Fri Jul 14 15:58:28.915345 2023] [:notice]
[pid 1:tid 140035002568128] ModSecuri>
Jul 14 08:58:28 ora super-web[21538]: [Fri Jul 14 15:58:28.925259 2023] [:notice]
[pid 1:tid 140035002568128] ModSecuri>
Jul 14 08:58:28 ora super-web[21538]: [Fri Jul 14 15:58:28.925323 2023] [:notice]
[pid 1:tid 140035002568128] ModSecuri>
Jul 14 08:58:28 ora super-web[21538]: [Fri Jul 14 15:58:28.925377 2023] [:notice]
[pid 1:tid 140035002568128] ModSecuri>
Jul 14 08:58:28 ora super-web[21538]: AH00558: httpd: Could not reliably determine
the server's fully qualified domain >
```

```
Jul 14 08:58:29 ora super-web[21538]: [Fri Jul 14 15:58:29.028048 2023] [ssl:warn]
[pid 1:tid 140035002568128] AH01909:>
Jul 14 08:58:29 ora super-web[21538]: [Fri Jul 14 15:58:29.028450 2023]
[lbmethod_heartbeat:notice] [pid 1:tid 14003500>
[user@ora ~]$ curl localhost:12345
Hello World!
[user@ora ~]$ systemctl --user stop super-web.service
[user@ora ~]$ curl localhost:12345
curl: (7) Failed to connect to localhost port 12345: Connection refused
[user@ora ~]$ systemctl --user start super-web.service
[user@ora ~]$ curl localhost:12345
Hello World!
[user@ora ~]$ systemctl --user status super-web.service
● super-web.service - Podman container-super-web.service
      Loaded: loaded (/home/user/.config/systemd/user/super-web.service; disabled;
vendor preset: disabled)
      Active: active (running) since Fri 2023-07-14 08:59:02 PDT; 8s ago
      Docs: man:podman-generate-systemd(1)
      Process: 21817 ExecStart=/usr/bin/podman start super-web (code=exited,
status=0/SUCCESS)
   Main PID: 21841 (conmon)
      CGroup:
/user.slice/user-1000.slice/user@1000.service/app.slice/super-web.service
             ├─21826 /usr/bin/slirp4netns --disable-host-loopback --mtu=65520
--enable-sandbox --enable-seccomp --enabl>
             ├─21828 rootlessport
             ├─21834 rootlessport-child
             ├─21841 /usr/bin/conmon --api-version 1 -c
933d6009ca7a4364539acbecf3a4381bf3856678c099b13cbe70966ec2c2b9e>
             ├─21844 httpd -D FOREGROUND
             ├─21872 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21873 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21874 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21875 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─21876 httpd -D FOREGROUND
             ├─21877 httpd -D FOREGROUND
             ├─21878 httpd -D FOREGROUND
             └─21879 httpd -D FOREGROUND


Jul 14 08:59:02 ora super-web[21841]: AH00558: httpd: Could not reliably determine
the server's fully qualified domain >
…
[user@ora ~]$ systemctl --user enable super-web.service
Created symlink
/home/user/.config/systemd/user/default.target.wants/super-web.service →
/home/user/.config/systemd/user/super-web.service.
[user@ora ~]$ exit
logout

PS C:\Users\User\Desktop> wsl --shutdown
```

```
Wait for 10 seconds
PS C:\Users\User\Desktop> wsl
[user@ora Desktop]$ cd
[user@ora ~]$ systemctl --user status super-web
● super-web.service - Podman container-super-web.service
       Loaded: loaded (/home/user/.config/systemd/user/super-web.service; enabled;
vendor preset: disabled)
       Active: active (running) since Fri 2023-07-14 09:02:35 PDT; 16s ago
       Docs: man:podman-generate-systemd(1)
       Process: 94 ExecStart=/usr/bin/podman start super-web (code=exited,
status=0/SUCCESS)
   Main PID: 151 (conmon)
       CGroup:
/user.slice/user-1000.slice/user@1000.service/app.slice/super-web.service
             ├─137 /usr/bin/slirp4netns --disable-host-loopback --mtu=65520
--enable-sandbox --enable-seccomp --enable->
             ├─139 rootlessport
             ├─144 rootlessport-child
             ├─151 /usr/bin/conmon --api-version 1 -c
933d6009ca7a4364539acbecf3a4381bf3856678c099b13cbe70966ec2c2b9e2 >
             ├─153 httpd -D FOREGROUND
             ├─183 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─184 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─185 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─186 /usr/bin/coreutils --coreutils-prog-shebang=cat /usr/bin/cat
             ├─187 httpd -D FOREGROUND
             ├─188 httpd -D FOREGROUND
             ├─189 httpd -D FOREGROUND
             └─190 httpd -D FOREGROUND

Jul 14 09:02:36 ora super-web[151]: [Fri Jul 14 16:02:36.167954 2023] [:notice] [pid
1:tid 139956160568768] ModSecurity>
…
Jul 14 09:02:36 ora super-web[151]: AH00558: httpd: Could not reliably determine the
server's fully qualified domain na>
…
Jul 14 09:02:36 ora super-web[151]: [Fri Jul 14 16:02:36.259188 2023]
[lbmethod_heartbeat:notice] [pid 1:tid 1399561605>
[user@ora Desktop]$ curl localhost:12345
Hello World!
[user@ora Desktop]$

[user@ora Desktop]$ sudo loginctl enable-linger user
[sudo] password for user:
[user@ora Desktop]$ loginctl show-user user
UID=1000
GID=1000
Name=user
Timestamp=Fri 2023-07-14 09:02:34 PDT
TimestampMonotonic=17759642809
RuntimePath=/run/user/1000
```

```
Service=user@1000.service
Slice=user-1000.slice
Display=c1
State=active
Sessions=c1
IdleHint=no
IdleSinceHint=1689350554513637
IdleSinceHintMonotonic=17759548223
Linger=yes
[user@ora Desktop]$
```

Note:

podman-generate-systemd(1)

System Container vs User

- systemctl without --user
- NO loginctl enable-linger
- /etc/systemd/system instead of ~/.config/systemd/user