Simplifying container orchestration with Ansible and Podman

Fabio Alessandro "Fale" Locati
EMEA Principal Specialist Solutions Architect, Red Hat

Red Hat

TOC

About me

- Working in IT since 2004, mostly in operations roles
- Active in open source (e.g.: Fedora FESCo)
- Ansible user since 2013
- Author of 5 books, 4 of which on Ansible
- EMEA Principal Specialist Solution Architect for Ansible @ Red Hat

Why

Why not Kubernetes?

- Heavy infrastructure overhead
- Steep learning curve
- Operational complexity

Kubernetes shaped problems

- Provide CaaS to others
- Deployments horizontal autoscaling
- Container auto-placement

Automate containers with Ansible

What is Podman?

- A daemonless, rootless alternative to Docker
- Recently donated to the CNCF
- Key features
  - Compatible with Docker CLI
  - Native support for OCI containers
  - Native support for Kubernetes objects

Automation strategy

- Ansible modules for Podman:
  - containers.podman (29 modules + 3 plug-ins)
- Workflow Overview
  - Use Ansible to deploy and manage containers with Podman
  - Use Ansible to startup, shutdown, and updates the containers

Alternative approach

What is systemd?

- A system and service manager for Linux
- Controls system processes, services, and dependencies
- Replaces older init systems (SysV, Upstart)
- Interesting features
  - Manages long-running services efficiently
  - Supports dependency management and auto-restarts
  - Provides robust logging and monitoring with journald
  - Allows extensions for custom kind of resources
- Why Use systemd for container management?
  - Enables native service control for containers
  - Simplifies startup, shutdown, and auto-restart of containers

What is Quadlet?

- A systemd helper for Podman
- Simplifies systemd unit file creation for containers
- Allows easy deployment and management of containerized services
- Technically, Quadlet does not exists (anymore)

Quadlet key features?

- Uses declarative configuration for container management
- Supports auto-restarts and dependencies
- Enables seamless integration with systemd services

Why Quadlet?

- Removes complexity from managing Podman containers via systemd
- Reduces the need for manual unit file configurations
- Ideal for persistent containerized applications

Quadlet base example

```
[Container]
ContainerName=myservice
Image=docker.io/my/service:1.0.0

[Install]
WantedBy=default.target
```

Strategy

- Place a file
- Reload systemd daemons
- Start and enable daemon

Ansible code example

```
- name: Ensure the container launcher is up to date
  ansible.builtin.copy:
    src: myservice.container
    dest: /etc/containers/systemd/myservice.container
    owner: root
    group: root
    mode: '0644'
  register: systemd_daemons
  notify: Restart myservice
- name: Reload systemd daemons if needed
  ansible.builtin.systemd:
    daemon_reload: true
  when: systemd_daemons.changed
- name: Ensure services are started and enabled
  ansible.builtin.service:
    name: myservice
    state: started
    enabled: true

- name: Restart myservice
  ansible.builtin.service:
    name: myservice
    state: restarted
```

Dependencies

```
[Unit]
After=local-fs.target nebula.service
```

Environment variables

```
[Container]
Environment=SECRET_KEY=YOUR_SECRET_KEY
```

Port publishing

```
[Container]
PublishPort=8080:80/tcp
```

Volumes

```
[Container]
Volume=/opt/mysrv:/etc/myservice
```

SELinux

What is SELinux?

A mandatory access control (MAC) mechanism for Linux that enforces security policies beyond traditional discretionary access control (DAC).

Why use SELinux?

- Provides fine-grained access control
- Limits damage from vulnerabilities
- Enforces least privilege principles

Why use SELinux?

- Containers do not contain (Dan Walsh)
- SELinux is important
- You should use it
- No, really, go and enable it, NOW

Ansible code example

```
- name: Ensure MyService folder exist
  ansible.builtin.file:
    path: /opt/mysrv
    state: directory
    mode: '0755'
- name: Ensure SELinux enforces the right security on MyService folder
  community.general.sefcontext:
    target: '/opt/mysrv(/.*)?'
    seuser: system_u
    setype: container_file_t
    state: present
```

Volumes - Exclusive access (retag)

```
[Container]
Volume=/opt/mysrv:/etc/myservice:Z
```

Volumes - Exclusive access (no-retag)

```
[Container]
Volume=/opt/mysrv:/etc/myservice:z
```

Volumes - Share access

```
ls -alhZ /var/opt
total 44K
drwxr-xr-x.  9 root root system_u:object_r:var_t:s0                       4.0K Aug 11 08:07 .
drwxr-xr-x. 23 root root system_u:object_r:var_t:s0                       4.0K Jan 17 19:48 ..
drwxr-xr-x.  7 root root system_u:object_r:container_file_t:s0:c311,c949  12K Feb  3 00:00 mysrv1
drwxr-xr-x.  2 root root system_u:object_r:container_file_t:s0:c124,c614 4.0K Dec 27 16:41 mysrv2
drwxr-xr-x.  5 root root system_u:object_r:container_file_t:s0:c47,c514   4.0K Aug 11 08:43 mysrv3
```

Volumes - Share access

```
[Container]
SecurityLabelLevel=s0
```

No security!

Volumes - Share access

```
ls -alhZ /var/opt
total 44K
drwxr-xr-x.  9 root root system_u:object_r:var_t:s0                       4.0K Aug 11 08:07 .
drwxr-xr-x. 23 root root system_u:object_r:var_t:s0                       4.0K Jan 17 19:48 ..
drwxr-xr-x.  7 root root system_u:object_r:container_file_t:s0:c311,c949  12K Feb  3 00:00 mysrv1
drwxr-xr-x.  2 root root system_u:object_r:container_file_t:s0:c124,c614 4.0K Dec 27 16:41 mysrv2
drwxr-xr-x.  5 root root system_u:object_r:container_file_t:s0:c47,c514  4.0K Aug 11 08:43 mysrv3
drwx------.  2 root root system_u:object_r:container_file_t:s0           4.0K Jan 17 22:54 mysrv4
```

Volumes - Share access

```
[Container]
SecurityLabelLevel=s0:c47,c514
```

Volumes - Share access

```
ls -alhZ /var/opt
total 44K
drwxr-xr-x.  9 root root system_u:object_r:var_t:s0                       4.0K Aug 11 08:07 .
drwxr-xr-x. 23 root root system_u:object_r:var_t:s0                       4.0K Jan 17 19:48 ..
drwxr-xr-x.  7 root root system_u:object_r:container_file_t:s0:c311,c949  12K Feb  3 00:00 mysrv1
drwxr-xr-x.  2 root root system_u:object_r:container_file_t:s0:c124,c614 4.0K Dec 27 16:41 mysrv2
drwxr-xr-x.  5 root root system_u:object_r:container_file_t:s0:c47,c514  4.0K Aug 11 08:43 mysrv3
drwx------.  2 root root system_u:object_r:container_file_t:s0           4.0K Jan 17 22:54 mysrv4
drwxr-xr-x.  6 root root system_u:object_r:container_file_t:s0:c47,c514  4.0K May  5  2024 mysrv5
```

Kubernetes objects

Kubernetes objects

```
[Install]
WantedBy=default.target

[Kube]
# Point to the yaml file in the same directory
Yaml=mySrv.yml
```

Kubernetes objects

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: haproxy
spec:
  containers:
    - name: haproxy
      image: docker.io/haproxytech/haproxy-alpine:3.1.1
      ports:
        - containerPort: 8448
          hostPort: 8448
        - containerPort: 443
          hostPort: 443
      volumeMounts:
        - mountPath: /usr/local/etc/haproxy
          name: config-volume
  volumes:
    - name: config-volume
      hostPath:
        path: /opt/haproxy
        type: Directory
```

Wrapping up

Wrapping up

- Kubernetes is good for Kubernetes-shaped problems
- Ansible and Podman can be great to run containers
- Ansible and Podman is a very straightforard solution

Questions?

Email: fale@redhat.com
Fediverse: @fale@fale.io

Red Hat

Links

- https://podman.io/docs/
- https://podman.io/blogs/2023/04/quadlet-tutorial.html
- https://docs.ansible.com/ansible/latest/
- https://fale.io/blog/2023/12/31/share-volumes-between-podman-systemd-services