systemd-nspawn
kontenery bliższe sercu administratora

Radosław Kujawa – radoslaw.kujawa@osec.pl

OSEC

28 listopada 2016

Kontenery dziś

- Ogromne ułatwienie procesu developmentu oraz testowania.
- Potencjalne utrudnienie w procesie wdrożenia na produkcje oraz późniejszym zarządzaniu.
- Kontener - zunifikowany sposób dostawy aplikacji, od developera do administratora?
- Specyfikacja kontenera jako elementu budulcowego systemu dalej nie rozwiązuje problemów komunikacyjnych.
- Docker budzi grozę wśród wielu osób zarządzających infrastrukturą.
- Konteneryzacja bardziej przyjazna adminom – `systemd-nspawn` – zademonstrujmy jak prosta i szybka jest!

Kontenery systemd-nspawn

- Oparte o mechanizm namespace.
- Dobrze zintegrowane z systemd.
- Chronione SELinuxem.
- Korzystające z narzędzi hosta do instalacji kontenerów, a także zarządzania nimi (`machinectl`, `systemctl` etc.).
- Domyślnie korzystają z systemu plików hosta, sieci hosta, oraz współdzielą namespace użytkowników (ale jest możliwość wydzielenia).
- Istnieje możliwość uruchamiania binarów innej architektury niż hosta (np. przez `systemd-binfmt`).

Instalacja

- Dostępne od Fedora 19.
- Zalecane systemd 231 lub nowsze – Fedora 25.
- Paczka `systemd-container` dostarcza potrzebne narzędzia.
- To wszystko! ☺

Instalacja kontenera

- Fedora:
  - `dnf --releasever=25 --installroot=/var/lib/machines/foo/ install systemd passwd dnf fedora-release`
  - Uwaga: cache `dnf` jest *wewnątrz* kontenera...
- Debian
  - `debootstrap --arch=amd64 stable /var/lib/machines/bar`
- Z wykorzystaniem innych narzędzi specyficznych dla danej dystrybucji.
- Istnieje możliwość instalacji innej dystrybucji niż system hosta wewnątrz kontenera!

SELinux

- Typ `container_t` dla procesów oraz `continer_file_t` i `container_ro_file_t`.
- `semanage fcontext -a -t container_file_t '/container(/.*)?'`
- `restorecon -R /container/`
- Możliwa bardziej zaawansowana polityka, wykraczająca poza typy `container*`.

Start kontenera

- Ustawienie hasła roota.
  - `systemd-nspawn -D /var/lib/machines/foo`
- Start kontenera.
  - `systemd-nspawn -bD /var/lib/machines/foo` (start interaktywny)
  - `systemd-nspawn -D /var/lib/machines/foo /opt/app` (uruchomienie polecenia w kontenerze)
  - Tworzy jednostkę systemd `machine-nazwa.scope` (transient).
  - Jest także widoczny w `machinectl -l`.

Korzystanie z gotowych kontenerów

- Import kontenera: `machinectl import-tar / import-raw`.
- Import kontenera z sieci: `machinectl pull-raw Fedora-Cloud-Base-25-1.3.x86_64`.
- Listowanie obrazów: `machinectl list-images`.

Start kontenera wraz ze startem hosta

- `systemctl enable machines.target`.
- `systemctl enable systemd-nspawn@foo.service`.
- Dostosowanie jednostki `systemd-nspawn@.service`.
- ... lub `/etc/systemd/nspawn/foo.nspawn`.
- Możliwe oczekiwanie przez hosta na start kontenera, celem budowania zależności (`--notify-ready`).

Zarządzanie z poziomu hosta

- `machinectl status kontener`.
- `systemctl -M kontener`.
- `journalctl -M kontener`.
- Procesy kontenera są widoczne w systemie hosta (dodatkowe pole `machine` w `ps`.

Zarządzanie storage kontenera

- System plików root kontenera – domyślnie osobny dla każdego.
- Rozwiązaniem BTRFS.
  - `--template`.
  - Tryb „ephemeral” (`-x`).
- Mount root ro.
- Dostęp do katalogów hosta z poziomu kontenera: `--bind`.

Zarządzanie siecią

- Wydzielenie interfejsu hosta dla kontenera: `--network-interface`.
- Forward portów do kontenera `--port`.
- Wirtualny interfejs między hostem a kontenerem: `--network-veth`.
- Wirtualny interfejs kontenera do bridge hosta: `--network-bridge`.

Podsumowując...

- Mechanizmy dostarczane przez systemd są bardziej niskopoziomowe.
- Bardziej konfigurowalne z perspektywy administratora.
- Kontenery nie są mechanizmem bezpieczeństwa.
- Próba przerzucenia klasycznych metod administracji do kontenerów może nie być właściwym rozwiązaniem.
- `rkt` także używa `systemd-nspawn`... i jest kompatybilny z Kubernetesem.

Koniec...

Dziękuje!
Czy są pytania?