IVC

Infrastructures Virtuelles et Conteneurs

Legond-Aubry Fabrice

fabrice.legond-aubry@parisnanterre.fr

Infrastructures Virtuelles et Conteneurs

Principes Pratiques

Plan du Cours

Podman

buildah

Conteneur: création et déploiement

- Créer un conteneur peut se faire à la main
  - ✓ Lourd, long.
  - ✓ Voir TD (création partielle)
- Il existe des outils pour automatiser les déploiements et la création
  - ✓ Déploiement: podman, docker, vagrant
  - ✓ Création : dockers (une partie des outils), openshift, buildah

Conteneur: podman/buildah

- Couple buildah / podman
  - ✓ Podman → déploiment / gestion des conteneurs
    - ➢ https://podman.io/releases/
  - ✓ Buildah → construction des images OCI
    - ➢ https://buildah.io/
  - ✓ Liens github: https://github.com/containers
- Solution pour Conteneur "rootless" et "daemonless"
- **Note: podman peut aussi gérer des conteneurs "rootful"**

Conteneur: podman/buildah

Présentation

- Rappel : Solution pour Conteneur "rootless" et "daemonless"
  - ✓ Mitigation des vulnérabilités potentielles lors de l'exécution du conteneur et du contrôleur du conteneur
  - ✓ Permet aux utilisateurs de partager des machines puissantes en limitant les risques d'escalade de privilège
  - ✓ Meilleure isolation des conteneurs imbriqués (Container-In-Container)
  - ✓ Ex: Docker CVE-2014-9357, CVE-2014-3519, CVE-2017-1002101, CVE-2019-5736, CVE-2020-13295
  - ✓ Ne résiste pas aux failles de noyaux / drivers matériels
  - ✓ Voir la partie sécurité
- Note: "Docker v20.10+" supporte le "rootless" mais reste "daemonful"
- Note: "LXC" supporte le rootless depuis 2013 mais reste "daemonful"

Conteneur: podman
Présentation

- Podman a un avantage :
  - ✓ Il est particulièrement compatible avec docker dans ses commandes
  - ✓ Alias docker='podman'
  - ✓ Basé sur le langage "go" (https://golang.org/)
- Gestionnaire sans deamon de conteneur (daemonless)
  - ✓ Contrairement à Docker (deamonful)
- Dépendances:
  - ✓ Voir le package manager de la distribution linux (apt/yum/dnf/…)
  - ✓ Systemd/contained, slirp4netns (réseaux), fuse-overlay-fs
  - ✓ Noyaux : cgroup, namespace

Conteneur: podman

Présentation

- Il existe des bind python (pypodman)
- Un Conteneur ne peut accéder à un namespace d'un autre conteneur
- Création d'espaces de ressources partagées (pods)
  - ✓ Même principe que pour Kubernetes et ses Pods
  - ✓ Un groupe (pod) de conteneur peut partager les mêmes ressources

Conteneur: podman

Présentation. Podman ne fait pas …

- Le (re) démarrage automatique des conteneurs
  - ✓ Utilisation des initd (systemd) de l'OS hôte
- Swarm (inondations)
  - ✓ Utilisation de l'orchestrateur de Kubernetes
- Contrôle de l'état de santé des conteneurs
  - ✓ En dev … systemd ? Side-conteneur de monitoring ? Démon ?
- Support des Docker API
  - ✓ Pas prévu

Conteneur: podman

Outils podman

- Définition des images à déployer
  - ✓ Utilise les OCI (définition standard et opensources des images)
  - ✓ https://opencontainers.org/ et https://github.com/opencontainers
  - ✓ **Cet élément n'est pas spécifique à podman !**
  - ✓ **github → containers (podman/buildah) et opencontainer (OCI)**
- Mécanisme pour télécharger une image OCI à partir d'un registre
  - ✓ https://github.com/containers/image
  - ✓ Ne permet pas la gestion des images dans un registre
- Outils pour déployer une image sur un système de fichier virtuel COW
  - ✓ https://github.com/containers/storage
  - ✓ Rappel système: COW = Copy-On-Write
- Outils pour exécuter un conteneur (lancement)
  - ✓ Spécification OCI pour l'exécution
  - ✓ https://opencontainers.org/release-notices/v1-0-2-runtime-spec

Conteneur: podman

Outils podman

- Outils pour exécuter un conteneur (lancement)
  - ✓ Spécification OCI pour l'exécution
  - ✓ https://opencontainers.org/release-notices/v1-0-2-runtime-spec
  - ✓ Outils runc (de la spec OCI). Même outil que pour docker.
- Un moyen standardisé de fixer / configurer le réseau
  - ✓ "Container Networking Interface"
  - ✓ Et outils réseaux classiques (bridge, …)
- Outils pour surveiller les conteneurs
  - ✓ Habituel. Commun. Outils d'administrations, …
- Un outil client (compatible pour les paramètres avec docker)
  - ✓ https://github.com/containers/podman
- **PAS DE SERVEUR (daemonless)**

Conteneur: podman

Commandes: les aides

- La base.
  - ✓ RTFM (http://docs.podman.io/en/latest/).
  - ✓ "man podman"
  - ✓ "podman –help" ou "podman commande --help"

- Pour les manipulations, voir le TD podman/buildah

- "podman info"
  - ✓ Informations sur le système hôte et la configuration podman

Conteneur: podman

Commandes: qqs manipulations d'images

- Note:
  - ✓ Les images ont un nom et un identifiant IMAGE_ID
    - ➢ On note l'utilisation de l'un ou l'autre par **IMAGE**
  - ✓ Note: Les conteneurs (exécution d'une image) ont un nom et un identifiant CONTAINER_ID
    - ➢ On note l'utilisation de l'un ou l'autre par **CONTAINE**R
- "podman images"
  - ✓ Liste des images stockées localement
- "podman image search texte" ou "podman search texte"
  - ✓ Chercher une image contenant 'texte'
  - ✓ "podman search httpd --filter=is-official"
- "podman image pull nom_image" ou "podman pull nom_image"
  - ✓ Charger une image localement, elle apparait avec "podman images"
- "podman import" permet d'importer le tgz d'un système et d'en faire un conteneur

Conteneur: podman

Commandes: exécution d'images

- "podman run –d –t IMAGE"
  - ✓ Exécution d'un conteneur et renvoie son id
  - ✓ Exécution sans état persistent (éphémère) par défaut
  - ✓ Si vous sortez (exit) le conteneur meurt
  - ✓ Détachement : ctrl+p puis ctrl+q pour laisser le conteneur en vie (detach). Ne fonctionne qu'avec les options "-t" and "-i"
- "podman start IMAGE"
- Rappel: les conteneurs sont identifiés par des résultats de fonctions de hash soit par un nom. Ex: CONTAINER_ID peut être
  - ✓ 9f93ce41fdc6d6ff98d02bdb3c68a7a4ebcf28f8627201e63f8ec17ebe47fa7e
  - ✓ 9f93ce41fdc6
  - ✓ Un nom est autogénéré si l'option "--name" n'est pas utilisé
  - ✓ Un nom est attribué si on utilise "podman run --name"

Conteneur: podman
Commandes: attachement / connexion

- Visible avec "podman container list" ou "podman ps -a"
  - ✓ Liste des conteneurs actifs
- "podman inspect CONTAINER"
  - ✓ Infos sur un conteneur (en json). → voir les répertoires
  - ✓ Rappel: un conteneur est une exécution d'image
- "podman attach CONTAINER"
  - ✓ Ouvrir une session tty sur le conteneur
  - ✓ Si vous sortez (exit) le conteneur meurt
  - ✓ Détachement : ctrl+p puis ctrl+q. peut être change avec l'option "--detach-keys"
- "podman exec -i -t CONTAINER bash"
  - ✓ Exécute une commande sur un conteneur en exécution
  - ✓ le "-t -i" permet d'ouvrir une session interactive
  - ✓ A la sortie du terminal le conteneur continue de s'exécuter

Conteneur: podman
Exercice 1 : Manipulations

- Création de vos premiers conteneurs
- Tester l'attachement / détachement
- Tester les option de "run" et "exec"
- Comparer les versions des programmes dans et hors conteneur
  - ✓ Dnf, gcc, …
  - ✓ Structure des fs
  - ✓ Processus
  - ✓ Identités (uid/gid)
  - ✓ Rechercher dans les cgroup les éléments qui pourraient être pour les conteneurs. Regarder les infos

Conteneur: podman

Commandes: gestion des conteneurs

- "podman container" et "podman system"
  - ✓ Gestion des conteneurs
- "podman container commit CONTAINER nom_nvl_image"
  - ✓ Création d'une nouvel image à partir de l'état d'une image en cours d'exécution
- Manipulation des IO des conteneurs
  - ✓ "podman unshare" → gestion des espaces de nommage
  - ✓ "podman network", "podman port" → réseau
  - ✓ "podman mount", "podman umount" → point de montage
  - ✓ "podman volume" → espace disque partagé entre l'hôte et le conteneur

Conteneur: podman

Commandes: gestion des conteneurs

- Gestion des pod (ensemble de conteneurs) : play / pod / generate

- Effacement :
  - ✓ "podman kill CONTAINER" → tuer un conteneur
  - ✓ "podman rm CONTAINER" → effacer définitivement
  - ✓ "podman rmi IMAGE" → effacer une image

Conteneur: podman

Exercice 2 : création d'un conteneur avec ssh

- Faire un conteneur à partir d'une fedora
  - ✓ Télécharger l'image dans votre dépôt local
- Créer un conteneur nommé "systemd" avec les éléments suivant
  - ✓ Délégation de port 2222 (ext) → 22 (int)
  - ✓ Limiter la mémoire à 512mo de RAM
  - ✓ Lancer "systemd" au démarrage du conteneur
- Se connecter au conteneur "systemd"
  - ✓ Installer les packages suivant
    - ➢ systemd, passwd, procps, vim, coreutils, ssh
  - ✓ Activer le service ssh dans systemd du conteneur
  - ✓ Ajouter un utilisateur avec son mot de passe
  - ✓ Tester la connexion

Conteneur: podman

Exercice 3 : Conteneur avec limitations

- Constater les différences pour les processus (ex: systemd / sshd)
  - ✓ les ids et les pids des processus entre l'hôte et l'invité (conteneur)
  - ✓ La visibilité des processus du conteneur et de l'hôte
- Copier "mon_app" dans le conteneur systemd
- Exécuter l'application "mon_app" dans l'environnement contraint
  - ✓ Vérifier le comportement du programme
- Exécuter le programme en même temps que la commande "htop"
- Tenter de changer la limite de RAM à partir du FS cgroupv2

Conteneur: buildah

Création de conteneur

- Au lieu de faire les manipulations d'installation des packages à la main
  - ✓ On souhaite automatiser
- Automatisation de la construction des images
  - ✓`buildah !!!!`
- Buildah utilise beaucoup de commandes similaires à podman
  - ✓ Commandes similaires rm, rmi, images; inspect, info, push, pull

Conteneur: buildah

Création de conteneur à partir d'une base vide

- Création du support de base à partir de rien
  - ✓ "tempName=$(buildah from scratch)"
  - ✓ "build run $tempScratchName bash"
  - ✓ Essayer de se connecter et constater
  - ✓ Monter l'image du conteneur
- Impossible → il faut recréer à la main le système de fichier
  - ✓ À la main comme pour le chroot
  - ✓ Avoir un tgz
  - ✓ Avoir une image OCI
- Aussi compliqué quasiment que pour le chroot

Conteneur: buildah

Exercice 4:

Création de conteneur à partir d'une base non vide

- Création du support de base à partir d'une image existant
  - ✓ "tempName=$(buildah from fedora)"
  - ✓ "build run $tempName bash"
- Créer un fichier d'initialisation ./monScriptInit.sh
  - ✓ Qui installe sshd, active sshd dans systemd, le démarre
- Configurer et installer le script d'initialisation
  - ✓ Utiliser la commande "config" de "buildah" pour fixer l'auteur, et le créateur"
  - ✓ Utiliser les commandes "copy", puis "config" pour fixer le script de démarrage

Conteneur: buildah

Exercice 4:

Création de conteneur à partir d'une base non vide

- Tester avec podman
  - ✓ Utiliser la "commit" de "buildah" pour créer une image nommée "fedora-with-ssh" (pour faire une image non définitive)
  - ✓ Utiliser podman pour tester l'image temporaire validée
- Si vous êtes satisfait, vous pouvez
  - ✓ Inspecter l'image (commande "inspect")
  - ✓ Démonter l'image (commande "unmount") et valider (commande "commit" à nouveau)
- Constater la présence de l'image via "buildah" et "podman".
- Effacer le conteneur de travail avec "buildah rm"

buildah bud -f Dockerfile -t fedora-httpd .

Conteneur: buildah

Exercice 5 :

Création de conteneur à partir d'un fichier Docker

- On peut utiliser la commande :
  - ✓"buildah build-using-dockerfile -f Dockerfile -t fedora-httpd ."
  - ✓"buildah bud -f Dockerfile -t fedora-httpd .
- Il faut créer un fichier de déploiement Docker
  - ✓https://docs.docker.com/engine/reference/builder/
- Exemple :

```
# Se baser sur la derniere Fedora
FROM fedora:latest
… A FAIRE …
```

Conteneur: buildah

Exercice 6 : création d'un conteneur de registre

- Création de la base (au choix) :
  - ✓ "buildah from registry"
  - ✓ "buildah from docker-daemon:registry:latest"
- Liste des images en conteneur
  - ✓ "buildah containers"
- Exécuter le registre
  - ✓ Note: le push/pull sur des registres publics nécessite des gestions d'authentification
  - ✓ Note: podman/buildah sont tatillons → utilisé l'option "--tls-verify=false"
- Lister les images (normalement vide)
- Créer une image simple et faire un push de limages dans votre registre
  - ✓ Vérifier l'OCI avec l'outil skopeo
- Effacer l'image créé de votre registre local
- Re-télécharger votre image à partir de votre registre

Conteneur: podman

Exercice 7 : des Pods avec podmann