Rapport Projet OS

Container avec Namespaces et Cgroup

Année universitaire
2020/2021

Réalisé par :
KHYATTI Ali
ELGHAZI Ilyass

Encadré par :
Prof. AIRAJ Mohammed

Introduction

Les conteneurs sont une forme de virtualisation du système d'exploitation. Un seul conteneur peut être utilisé pour exécuter n'importe quoi, d'un petit microservice ou processus logiciel à une application plus grande. À l'intérieur d'un conteneur se trouvent tous les exécutables, le code binaire, les bibliothèques et les fichiers de configuration nécessaires. Cependant, par rapport aux approches de virtualisation de serveurs ou de machines, les conteneurs ne contiennent pas d'images de système d'exploitation. Cela les rend plus légers et portables, avec beaucoup moins de frais généraux.

Léger: les conteneurs partagent le noyau du système d'exploitation de la machine et ne nécessitent donc pas de système d'exploitation par application, ce qui améliore l'efficacité des serveurs et réduit les coûts de serveur et de licence

Namespace :

Les espaces de noms Linux comprennent certaines des technologies fondamentales derrière la plupart des implémentations de conteneurs modernes. À un niveau élevé, ils permettent l'isolement des ressources système globales entre des processus indépendants. Par exemple, l'espace de noms PID isole l'espace de numéros d'ID de processus. Cela signifie que deux processus exécutés sur le même hôte peuvent avoir le même PID!

Ce niveau d'isolation est clairement utile dans le monde des conteneurs. Sans espaces de noms, un processus s'exécutant dans le conteneur A pourrait , par exemple, démonter un système de fichiers important dans le conteneur B, ou modifier le nom d'hôte du conteneur C, ou supprimer une interface réseau du conteneur D. Avec les namespaces vous ne savait même pas que les processus dans les conteneurs B, C et D existent, Il s'ensuit que vous ne pouvez pas interférer avec quelque chose si ce n'est pas visible pour vous. ces espaces fournir aux processus leur propre vision du système. les espaces de noms limitent ce qu'on peut voir (et par conséquent, ce que vous pouvez utiliser).

Ces espaces de noms sont disponibles dans les noyaux modernes:

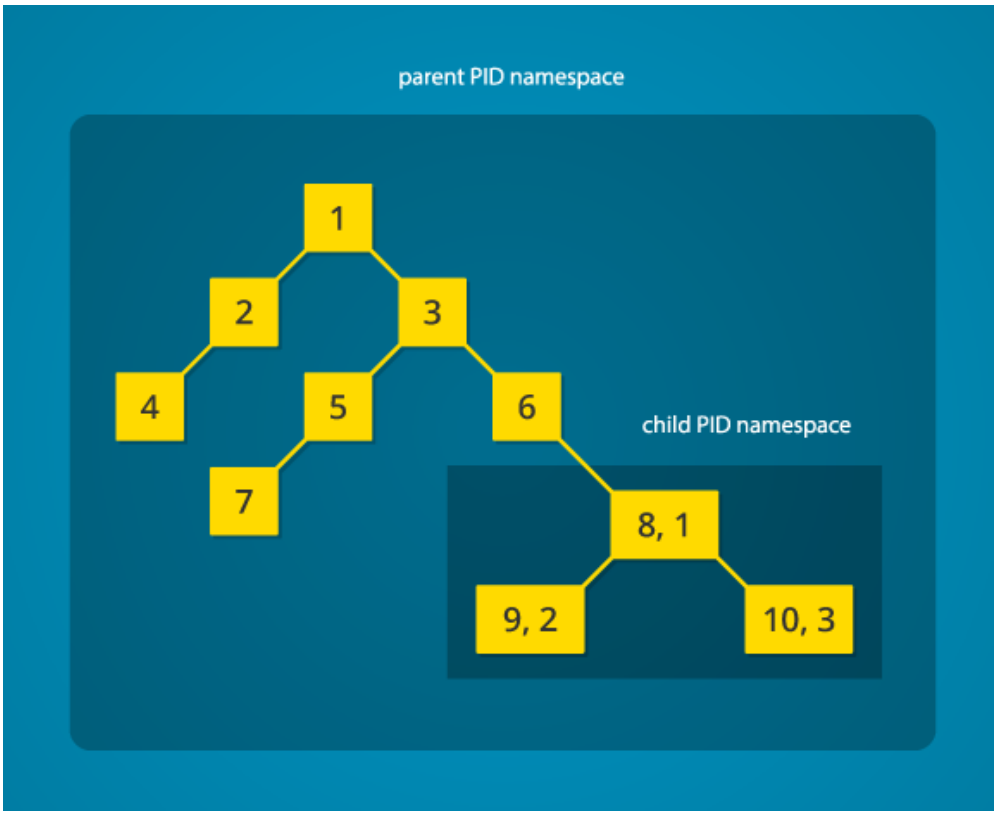
- Espace de noms réseau (NET)
- Monter (MNT)
- Espace de noms UTS ou Hostname
- ID de processus ou espace de noms PID
- Communication inter-processus ou espace de noms IPC
- Espace de noms utilisateur
- Espace de noms cgroup

(Nous allons les détailler individuellement.)

Chaque processus appartient à un espace de noms de chaque type.

Les espaces de noms sont créés avec deux méthodes:

- le `clone()` appel système (utilisé lors de la création de nouveaux threads et processus),
- le `unshare()` appel système.



Groupe de contrôle (cgroup) :

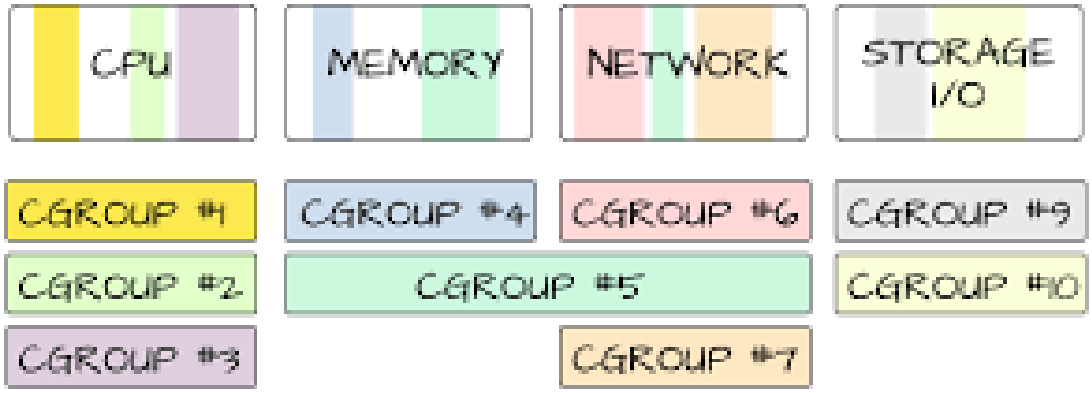
Un * cgroup * associe un ensemble de tâches à un ensemble de paramètres pour un ou plusieurs sous-systèmes.

Un * sous-système * est un module qui utilise le regroupement de tâches et les installations fournies par les groupes de contrôle pour traiter des groupes de tâches de manières particulières. Un sous-système est généralement un «contrôleur de ressources» qui planifie une ressource ou applique des limites par un groupe de contrôle, mais cela peut être tout ce qui veut agir sur un groupe de processus, par exemple un sous-système de virtualisation.

À tout moment, il peut y avoir plusieurs hiérarchies de tâches actives cgroups. Chaque hiérarchie est une partition de toutes les tâches du système.

Les groupes de contrôle nous fournissent la mesure et la limitation des ressources.

- Cela couvre un certain nombre de "suspects habituels" comme:
    - Mémoire
    - CPU
    - bloc E / S
    - réseau (avec la coopération des tables ip/tc)
- Et quelques exotiques:
    - énormes pages (une façon spéciale d'allouer de la mémoire)
    - RDMA (ressources spécifiques à InfiniBand / transfert de mémoire à distance



Source Code (container.c)

```c
char *hostname;
int child(void *args)
{
    // char hostname[20] = "KHYATTI&ELGHAZI";

    printf("child pid inside: %d.\n", getpid());
    printf("parent pid inside: %d.\n", getppid());
    sleep(1);
    printf ("running container ... \n");
    char cwd[255];
    if(getcwd(cwd, sizeof(cwd))!= NULL){
        sethostname(hostname,sizeof(hostname));
        printf ("processing...");
        printf ("cwd = %s \n\n", cwd);
        chroot(cwd);
        chdir("/");
        mount("proc", "/proc", "proc", 0, NULL);
        sleep(1);

        printf ("Child (new) network Namespace ... \n");
        system("ip link");
        sleep(1);

        printf("\n\n\nexecuting new bash ...\n");
        execlp("/bin/bash", "/bin/bash" , NULL);
```

Cette fonction présente le call-back de notre méthode clone qui va être appelé lors de la création d'un nouveau processus isolé, dans laquelle on va attribuer un nouveau hostname de notre processus, puis on change la racine au répertoire courant, après ça on attache le répertoire (kernel) /proc au répertoire proc de la nouvelle racine, en fin on peut exécuter notre nouveau bash.

```
int main(int argc, char const *argv[])
{
    int flags = CLONE_NEWUTS|CLONE_NEWPID|CLONE_NEWIPC|CLONE_NEWNS|CLONE_NEWNET;
    char cmd[200];

    hostname = malloc(30*sizeof(hostname));
    strcpy(hostname, argv[1]);

     printf("Original (parent) Network Namespace:\n");
     system("ip link");
     printf("\n\n");



    pid_t pid = clone(child, malloc(4096) + 4096, SIGCHLD|flags, NULL);
    if (pid == -1) {
        perror("clone");
        exit(1);
    }

    printf("\nChild pid outside: %d.\n\n", pid);

    system(" cgcreate -g memory:KH_GH_shell");

    system(" echo 50M > /sys/fs/cgroup/memory/KH_GH_shell/memory.limit_in_bytes");
```

Dans cette partie du code vous permet de créer un processus isolé en utilisant la méthode clone(), a son propre vue différent du processus qui celui en dehors, grâce à la création du nouveau cgroup avec limitation de mémoire à 50M dans cet exemple et lui associé ce nouveau processus crée. Voici la signification des namepaces utilisée dans la méthode clone

- CLONE_NEWNET: créer un nouveau espace de nom réseau.
- CLONE_NEWUTS: créer un nouveau espace de nom partage le m identificateurs et NIS domaine.
- CLONE_NEWIPC: créer un nouveau espace de nom qui agit l'isolation de la communication inter processus.
- CLONE_NEWPID: créer un nouveau espace de nom PID.
- CLONE_NEWNS: créer un nouveau espace de nom de montage